UNIVERSITY OF KWAZULU-NATAL

A model for DevOps implementation in South Africa

Mudaray Marimuthu

9261483

A thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy

School of Management, IT and Governance

College of Law and Management Studies

Supervisors: Dr S Ranjeeth

Prof I Govender

2024

ACKNOWLEDGEMENTS

- I would like to express my sincere gratitude to my supervisors, Dr Sanjay Ranjeeth and Prof Irene Govender, for their unwavering support, guidance, and mentorship.

- I extend my appreciation to UCDP for funding my data collection.

- I am thankful to all the participants who sacrificed their time to afford me an interview or complete my questionnaires. Not forgetting my ex-students who assisted in identifying DevOp Professionals to participate in this study.

- I wish to thank my wife, Anusha, and my children, Kadhir and Kulvir, for all their love and support during this journey and beyond.

LIST OF ABBREVIATIONS

AHP	Analytical Hierarchy Process
ART	Agile Release Train
AVE	Average Variance Extracted
AWS	Amazon Web Services
BI	Behavioural Intention
CAMS Model	Culture, Automation, Measurement, and Sharing Model
CD	Continuous Automation Deployment
CDE	Continuous Delivery
CFA	Confirmatory Factor Analysis
CFI	Comparative fit index
CI	Continuous Integration
CM	Configuration Management
CPS	Cyber-Physical Systems
CR	Composite Reliability
CSF	Critical Success Factors
C-TAM-TPB	Combined TAM and TPB
DOI	Diffusion of Innovation
DTPB	Decomposed Theory of Planned Behaviour
EE	Effort Expectancy
EFA	Exploratory Factor Analysis
FC	Facilitating Conditions
IaC	Infrastructure as Code

HM	Hedonic Motivation
IAST	Interactive Application Security Testing
IID	Iterative and Incremental Development
IoT	Internet of Things
IS	Information Systems
IT	Information Technology
KIS	Keep it Simple
KMO	Kaiser-Meyer-Olkin
MM	Motivation Model
PAF	Principal Axis Factoring
PCB	Perceived Behavioural Control
PCI	Perceived Characteristics of Innovating
PE	Performance Expectancy
PEOU	Perceived Ease of Use
PU	Perceived Usefulness
PNFI	Parsimony Normed Fit Index
PPP	Phased Program Planning
RMSEA	Root mean square error of approximation
ROI	Return on Investment
SAFe	Scaled Agile Framework
SAST	Static Application Security Testing
SCT	Social Cognitive Theory
SD	Standard Deviation
SDM	Software Development Methodologies
SEM	Structural Equation Modelling
SI	Social Influence

SNAC Framework	Stakeholders, Needs, Alterable and Constraints Framework
SPM	Software Process Model
SRMR	Standardised root mean square residual
TAM	Technology Acceptance Model
TDD	Test Driven Methodology
TIB	Theory of Interpersonal Behavior
TOSCA	Topology and Orchestration Specification for Cloud Applications
TPB	Theory of Planned Behaviour
TRA	Theory of Reasoned Action
UI	User Interface
UML	Unified Modelling Language
UP	Unified Process
US	United States
UTAUT	Unified Theory of Acceptance and Use of Technology
XP	Extreme Programming
XSS	Cross-Site Scripting

ABSTRACT

Software is a vital component of a digital economy. Organisations use software for operational efficiency, to gain competitive advantage and to retain customers. Software systems have become the focal point for competitive advantage and organisations have embraced software development methodologies that have the capacity to handle changing user requirements in a quick and nondisruptive manner. The advent of agile methodologies ensured that there was a universally endorsed software development methodology that could deliver on the promise of speedy development while maintaining sufficient quality standards to ensure end-user satisfaction. However, the euphoria that surrounded the perception of success with regard to agile methodology was soon dispelled because of challenges related to the phases of software release and deployment into the functional ("live/operational") environment. Hence, the intended benefits of agile methodology became somewhat limited thereby compromising the methodology's acceptance on a universal platform. To overcome this impediment, the DevOps approach was introduced to obviate the disconnect between the development and deployment teams so that software features can be rapidly deployed.

The DevOps approach did, however, come with its own set of challenges. Software practitioners were challenged to acquire an understanding of the factors that achieve acceptance and enable DevOps success. Although these factors have been extensively researched in the context of agile methodology, there is a lack of empirical evidence that attests to the acceptance and success factors for DevOps implementation. The current study was undertaken to provide a resolution to this gap that has been identified in the body of knowledge pertaining to software development methodology. The study leverages comprehensive empirical evidence from the professional software development community in South Africa to enable the researcher to converge the study's analysis and discussion to a cogent model of DevOps acceptance and success factors.

From a research design perspective, a sequential exploratory research method consisting of qualitative and quantitative phases of research was employed in this study. The qualitative method used a phenomenological approach to examine DevOps professionals' lived experience of DevOps to validate success and acceptance factors that have been identified in the academic literature. The outputs from the qualitative phase of the study were integrated with existing literature to create a conceptual model that was then subjected to quantitative inquiry using the technique of structural equation modelling.

The results from the quantitative inquiry identified the factors of *automation, Devops tools and technology and organisational culture* as being critical to DevOps success. From a methodology acceptance perspective, *performance expectancy, effort expectancy and social influence* were identified as crucial determinants of the *behavioural intention to use DevOps*.

The outcome of this study is the generation of a concise list of factors that are structured along the dimensions of DevOps success and DevOps acceptance. These factors have been empirically validated from the academic and practitioner perspectives thereby providing practitioners with an ideal resource to understand DevOps usage and enhance DevOps adoption in the professional sector. It also provides sufficient academic content to enable researchers to use the knowledge generated in the study as a platform from which to sustain the evolution of agile methodology and DevOps knowledge.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION TO THE STUDY

1.1 INTRODUCTION

Software is the cornerstone of a digital economy. With the proliferation of mobile devices and fast internet connectivity, software plays a critical role in how people and businesses interact. Hence, software has become the fundamental building block for society from both the social and economic perspectives, enabling individuals to live a productive life spanning from gaming to social media and for businesses to compete, survive, and innovate. Therefore, individuals and businesses are always demanding greater functionality, quality and reliability from the software applications that they interact with.

Due to the fast-paced nature of the business world, businesses need to be more flexible and innovative when it comes to effective and continuous software delivery to stay competitive (Colavita, 2016). In many organisations, the processes of the software development department and the operations department, which handle the implementation and maintenance of software, are not aligned (Jonker, 2017; Wiedemann et al., 2020). Software development is a process of creating a software artefact that aids organisations in achieving their organisational goals. It normally entails requirements gathering, analysis, design, and programming (Pressman, 2010). Whereas Information Technology (IT) operations entail the deployment of this artefact within the operational environment, enabling end users to derive benefits from its use (John et al., 2021).

With agile development, the optimisation of the operational phase is overlooked. For this reason, amongst others, organisations that use agile methodology do not have the capacity to guarantee timely delivery and deployment of a software system (Jonker, 2017). DevOps was introduced as a possible solution to this problem. Patrick Debois, a highly acclaimed software engineer who worked together with both operations and development teams during various projects, realised that there was a better way to handle development and operations and the conflicts surrounding them, and this is where the DevOps movement began (Liu & Zhou, 2017). DevOps extends the concept of the agile approach by using Agile practices to increase collaboration between the operations and development departments (Wiedemann et al., 2019).

In response to the lack of continuous deployment of agile code, software development industries are moving to DevOps. DevOps is a process that bridges the gap between development and operations, focuses on error reduction when code is deployed, and more frequently releases code into the end user environment (Aiello & Sachs, 2016). For this to be successful, coordination and coherence among different software development teams must be coordinated. These teams are the development team, quality assurance team and operations teams. According to information systems (IS) practitioners, the benefits of this can be enormous, leading to technical, cultural, and business benefits (Jha et al., 2023).

The preceding discussion provides a context for the content that will be presented in the 1st chapter of this thesis. The chapter will conclude with a presentation of the research problem, the topic and the main cohort of questions that will guide the study from conception to completion. A brief discussion of the methodological design used to structure the study's empirical phases will also be provided. Additionally, the justification and contribution of the study to the body of knowledge in the domain of software development methodology will be highlighted. An explanation of the chapters that comprise this thesis is provided at the conclusion of this chapter.

1.2 BACKGROUND AND OUTLINE OF THE RESEARCH PROBLEM

Early software development was characterised by the use of prescriptive software process models. These models, also referred to as structured system development models, are appropriate when all requirements must be known before the development of the system can be initiated, together with all analysis and design models and decisions documented. Although these models were successful in developing mission-critical systems, they were not appropriate for business applications in the modern era (Pressman, 2010). Since business applications are frequently changing, software development methodologies that can quickly adapt and implement these changes are more suitable. Hence, to achieve this imperative, the software development community developed an Agile software development methodology. This methodology can be implemented using various software process models (e.g. Scrum and Extreme Programming) and is based on the Agile Manifesto that focuses on individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation and responding to change over following a

plan (Pressman, 2010). The Standish report of 2015 shows that 11% of the projects that followed the Waterfall approach were successful, and 33% of projects that followed the Agile process were successful. The Standish Group Chaos Study of 2020 further indicates that Agile initiatives are three times more likely to succeed than Waterfall projects (Standish, 2020). Although there is a significantly better success rate in Agile projects than in Waterfall, Agile projects did not deliver an optimally functioning system (Standish, 2015). Almeida et al. (2022) mention that the problems with Agile software development are the delay in the delivery of new features, incompatible software components, the product's quality being inadequately verified before release, and new functionalities disrupting existing operations. One of the reasons for this sub-optimal outcome is the lack of cohesion in ensuring the transition from software development to software "rollout" in what should be a continuous cycle (Gartner, 2015). Gartner (2015) further states that Agile methodologies tend to lead to a continual flow of new and modified software into the *operational environment,* thereby placing a burden on business governance and operations teams that results in significant changes to working practices.

According to Chen and Power (2015), the biggest challenge to Agile success is at the organisational level where the disparate business divisions work independently, thereby adding a level of additional complexity to the delivery and deployment of software systems into the working environment. Each business division has its own goals, methods of working and perceived areas of control, which results in tensions between divisions owing to competing goals (Chen & Power, 2015) This challenge to successful software development and deployment is magnified in the anecdotal observation by Chen and Power (2015, p. 53) in reference to an organisational setting: "…*we needed root access to the servers, and another team controlled this permission. Arriving at a solution involved much consultation and negotiation over six months*." To break down barriers among teams and promote a collaborative culture, leadership teams needed to restructure the organisation, and this imperative has been a catalyst for the emergence of the DevOps strategy.

1.3 PROBLEM STATEMENT

In response to the fluctuating economic environment, organisations strive to obtain a competitive advantage by leveraging the benefits provided by software systems. These systems are under constant pressure to enhance organisational processes and profitability and provide the end users with a satisfying experience. It is imperative to produce these systems in a timely manner, within budget, with minimal resource consumption, and to deliver on user expectations. The systems must also embrace changing user requirements and have high visibility to a client community early in the development cycle (Ajiga et al., 2024). In response to the preceding cohort of requirements, the software development methodology needed to align with a strategy that embraced dynamism rather than being too rigid and structured (Mishra & Alzoubi, 2023). It is within this context that the Agile software development methodology was created. Although this methodology led to the software's rapid development, the software's delivery speed to the operational environment has not been resolved. To improve this process, many companies are adopting DevOps to provide a seamless flow of software from creation to the final delivery into the operational environment, allowing users and businesses to benefit from the software increments sooner (Al Masud et al., 2022). As companies are transitioning into DevOps, it becomes important to understand the factors that drive the acceptance and success of DevOps. This will assist other organisations in improving their adoption process and ultimately improve the success of the software developed. Numerous studies (e.g., (Colavita, 2016; Gupta et al., 2017; Kamuto & Langerman, 2017; Lwakatare et al., 2016; Rowse & Cohen, 2021; Sánchez-Gordón & Colomo-Palacios, 2018)) have documented the need to incorporate a DevOps strategy into the software process. However, there is a dearth of empirical evidence attesting to the acceptance (Anderson, 2019) and success factors that are pivotal in ensuring the overall success of DevOps (Azad, 2022; Gupta et al., 2017).

A discernible trend that seems to exacerbate this problem is that “success factors” and “acceptance factors” studies have been confined to the academic sector. In contrast, practically implementable business solutions such as Agile methodology and DevOps have been largely confined to the practitioner sector. Hence, it is the intention of the current study to obviate the divide between academia and the professional software development sector as well as to develop an acceptance and success model that underpins the DevOps software development process to better inform industry practice and contribute to academic knowledge in the domain of software engineering.

With the advancement of technology, the world is moving to a digital economy known as Industry 4.0. This industry will be driven by big data, IoT and cloud computing. For Industry 4.0 to be successful, the continuous development and deployment of software are vital. According to Fitsilis et al. (2018), DevOps will play a critical role in this success. Wijaya et al. (2019) are also of the opinion that disruptive technologies such as smart homes, smart cars, smart cities, big data, Industry 4.0 and the Internet of Things (IoT) can significantly alter value chains and business models. Because of this rapidly changing environment, IT functions need to innovate constantly and keep abreast of new product developments. Therefore, there is a need to dismantle the traditional silos between development and operations in order to deliver products in a shorter cycle time.

Hence, knowledge of factors affecting acceptance and success in DevOps projects is necessary because understanding these factors can increase the successful adoption of DevOps and contribute to the effective and efficient development and deployment of software that is important in Industry 4.0.

Research on acceptance and success factors are covered in numerous studies where the main focus is on Agile software development (e.g., Abd et al., 2016; Chiyangwa & Mnkandla, 2017; Chow & Cao, 2008; Misra et al., 2010; Nguyen, 2016a; Stankovic et al., 2013). The studies have collected primary research data from respondents who were part of the software development process, i.e. individuals in the Agile teams consisting primarily of programmers. DevOps is much broader than this and includes quality assurance and operational stakeholders. Hence, Agile studies on success factors have modelled these factors from the developers' perspective only, and therefore, the findings from these studies cannot be aligned to the DevOps environment because of the additional stakeholders that form part of the DevOps team. The operations and quality assurance stakeholders are lacking from these studies and success factors cannot be assumed to be the same as is the case with DevOps. The main stakeholders in the domain of Agile methodology are the business analysts and the software engineers. However, DevOps has highlighted the need to obtain perspectives from stakeholders who are engaged with maintaining the technical infrastructure where the success factors are distinctly different. The preceding assertion forms the basis for the study's main research question, which is presented in the next section.

1.4 RESEARCH QUESTIONS

Based on the problem statement, the study's main research question is:

How can experiential knowledge of DevOps practitioners be used to develop an acceptance and success model to influence the usage of DevOps?

The solution to the study's main research question will be structured according to a set of sub-questions that read as follows:

1. What are South African software practitioners' perspectives on the DevOps phenomenon?
2. What acceptance factors influence software professionals' intention to use DevOps?
3. What success factors influence DevOps project success?
4. Which acceptance and success factors are critical to the adoption of DevOps?

1.5 RESEARCH OBJECTIVES

The research questions are aimed at achieving the following objectives:

1. To understand the definition of DevOps and its benefits and challenges from the lived experience of software professionals working within the DevOps environment.
2. To determine which factors influence software professionals' intention to use DevOps.
3. To determine success factors that influence DevOps project success.
4. To propose an acceptance and success factors model for the adoption of DevOps.

1.6 RESEARCH JUSTIFICATION

The realisation that software is an essential ingredient of the digital era has forced the software development community to continuously innovate the software development process to allow for more software releases in a shorter timescale, decrease the mean time between failures and increase quality. These attributes are deemed to be achievable, provided a DevOps approach is used. According to the 2017 State of DevOps report, DevOps teams have steadily increased from 16% in 2014 to 27% in 2017 globally. A survey by CA Technologies South Africa indicated that 94% of their respondents in South Africa indicated a greater need for DevOps, but only 44% indicated that they have a DevOps strategy in place (CA_Technologies, 2015). Furthermore, Rowse and Cohen (2021) reported that only 54% of the respondents in their South

African study indicated that they use the key DevOps practice of continuous deployment frequently. According to Gartner's 2018 hype cycle, DevOps is now positioned in the "trough of disillusionment", where interest wanes as implementations fail to deliver on the early promise. Hence, it is very important to study the real-world implications and challenges of DevOps to perhaps clarify some of the expectations and realities behind the approach. The current study has been crafted to meet the aforementioned imperative, which is envisaged to provide knowledge on the acceptance success factors that underpin the DevOps approach to software development.

1.7 CONTRIBUTION TO THE BODY OF KNOWLEDGE

This study will contribute to a better understanding of the acceptance and success factors of DevOps projects. As highlighted, numerous studies have been conducted on the success factors in software development. However, these studies are mainly focused on previous paradigms of development, the traditional waterfall process or the Agile process. Since DevOps is still in its infancy, apart from mainly practitioners' perspectives, there are only a limited number of academic studies that have been carried out on DevOps, especially on acceptance and success factors. Hence, the intention of the current study is to contribute to both theory and practice. The theoretical contribution of this study to the success factors literature is its extension of this literature to the context of DevOps in order to enrich knowledge and understanding of the success factors in DevOps projects. From the practice perspective, these factors can be used by industry practitioners who are going to implement DevOps, thereby enabling them to gain knowledge of the strengths and weaknesses of using a DevOps approach.

1.8 OVERVIEW OF METHODOLOGY

A mixed-method research approach is used to address the research questions and objectives (Creswell & Clark, 2007). This entails using a sequential research design consisting of qualitative and quantitative phases. The qualitative phase is used to understand the lived experience of DevOps by software professionals using phenomenology. The quantitative phase is used to provide a descriptive analysis of acceptance and success factors and to determine which of these factors significantly influence the acceptance and success of DevOps. The theoretical foundation for the study is guided by analysing the current literature on success factors and the existing theories on technology acceptance.

1.9 STRUCTURE OF THE THESIS

An outline of the study is presented in the form of chapters in the narrative that follows:

Chapter 2 – The Past, Present and Future of software development processes

This chapter will discuss the past, present and future of software development by comparing various process models. First, the code and fix model will be discussed, and its shortcomings in large software systems will be highlighted. Thereafter, prescriptive models are discussed together with their inability to meet the demand of the modern software development environment. To overcome this, Agile models are discussed, highlighting their ability to build software quickly but not deploy rapidly. This is where DevOps is discussed as a process that allows for the rapid development and deployment of code.

Chapter 3 - Theoretical Underpinning and the Study's Preliminary Conceptual Model

The conceptual model for the study is discussed. Using the current literature pertaining to success factors within the software development domain, various success factors are identified within the context of DevOps success. Furthermore, various acceptance theories are discussed, and the appropriate model for this study is identified and discussed. This chapter proposes a preliminary model to study DevOps acceptance and success factors.

Chapter 4 – Research Design and Methodology

A comparison of the qualitative, quantitative and mixed research designs is discussed in this chapter. An in-depth explanation of sequential research design is provided together with a justification of why it is the most appropriate design for this study. Other aspects of the methodology are also explained, such as the population, sampling techniques and size, data analysis and research validity.

Chapter 5 – Qualitative Data Analysis of the lived experience of DevOps professionals

The interview findings will be presented in this chapter. Analysis of interview data into themes will be facilitated using the qualitative data analysis tool named NVivo. Results will be presented in the following categories: definition of DevOps, benefits, challenges, success and acceptance factors. Content analysis will be used to aid this process.

Chapter 6 – Revised conceptual model and study's hypotheses

This chapter integrates the findings of the qualitative analysis with existing literature, resulting in a revised conceptual model and research hypotheses.

Chapter 7 – Quantitative Data Analysis

Quantitative responses will be presented in this chapter. Analysis is broken down into demographic analysis, descriptive analysis, correlation analysis, regression analysis and model testing using structural equation modelling. For the inferential statistics, normalization, reliability, factor analysis and multicollinearity were determined to ensure the validity of the findings. The revised conceptual model from Chapter 5 is validated in this chapter.

Chapter 8 – Discussion of findings

A discussion of the qualitative and quantitative results will be presented in this chapter. The triangulation of the qualitative and quantitative findings and the DevOps model for acceptance and success are also discussed.

Chapter 9 – Conclusion, research contributions and future work

This chapter will conclude by discussing how this study contributes to theory and practice. The limitations of the study and future studies are mentioned.

1.10 SUMMARY

This chapter provided a background of the challenges to rapid software deployment and how DevOps can overcome them. However, for DevOps to have its desired impact, it must be accepted and successful. According to the problem statement, there is a lack of research regarding DevOps acceptance and success factors, and therefore, there is a need to understand these factors so DevOps can be adopted successfully. The next chapter discusses various software process models and the impact of DevOps on the rapid development and deployment of software.

CHAPTER 2: THE PAST, PRESENT, AND FUTURE OF SOFTWARE DEVELOPMENT PROCESSES

2.1 INTRODUCTION

Hart (1998) defines a literature review as a succinct impartial exploration of research and non-research literature on the topic under investigation. This enables the researcher to provide the audience with the most recent literature on a subject, thereby achieving a goal that justifies the subject under study by highlighting the gaps in the current literature (Cronin et al., 2008; Denney & Tewksbury, 2013). These ideas are further substantiated by Creswell (2014), who states, "The literature in a research study accomplishes several purposes: It shares with the reader the results of other studies closely related to the study being investigated. It relates a study to the larger, ongoing dialogue in the literature about a topic, filling in gaps and extending prior studies (Cooper, 2010; Marshall & Rossman, 1989). It provides a framework for establishing the importance of the study." There are various types of literature review: traditional or narrative literature review, systematic literature review, meta-analysis and meta-synthesis (Cronin et al., 2008). This study followed a traditional or narrative literature review since the researcher wished to critically analyse and summarise existing literature, allowing for meaningful conclusions regarding the investigated topic. Hence, this approach leads to an enriched context of the current literature, highlighting the importance of the new research.

According to Hart (1998), a progressive "narrowing of the topic" approach can be used when generating a literature review. Leedy and Ormrod (2005) and Hofstee (2006) suggested a similar approach referring to these as an "inverted pyramid" and the "funnel method" (Hofstee, 2006, p. 95), respectively. Furthermore, Webster and Watson (2002) recommend following a topic-centric approach to generate better quality literature reviews in IS. According to Khoo et al. (2011), this approach results in a more critical review that allows the researchers to focus on the studied domain. Furthermore, Cooper (1988) suggests that literature reviews can be organised chronologically, conceptually or methodologically, with reviews also combining these organisations' methods to address the historical perspective.

The funnel, topic centric and chronological order methods were used to develop this literature review. The literature review discusses the earliest approach to software development. As software became more intricate and complex, there was a progression to more structured methodologies. With the advent of the Internet, there is a need for software to be developed rapidly to allow clients to get the benefits of this software as soon as possible. This chapter

reviews the processes used to create software, discussing these processes from past, present, and future perspectives. This chapter further explains the strengths and weaknesses of the different process models that underpin the DevOps process on which this study focuses. DevOps is situated within the domain of software development, and the researcher discussed previous software development process models to explain how DevOps fits into this process. The sequential diagram in Figure 2.1 illustrates the chronological flow of the literature review from the past to the present.

Furthermore, a funnel and topic-centric approach is also followed. Figure 2. 2 illustrates this in a hierarchical diagram displaying the essential topics discussed in this literature review.

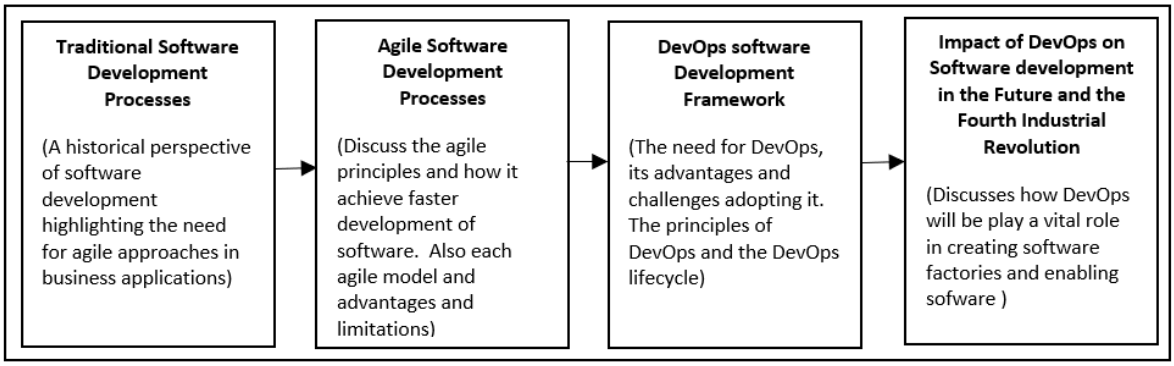

Figure 2.1: Sequential topic discussed in the literature review (Own Work)

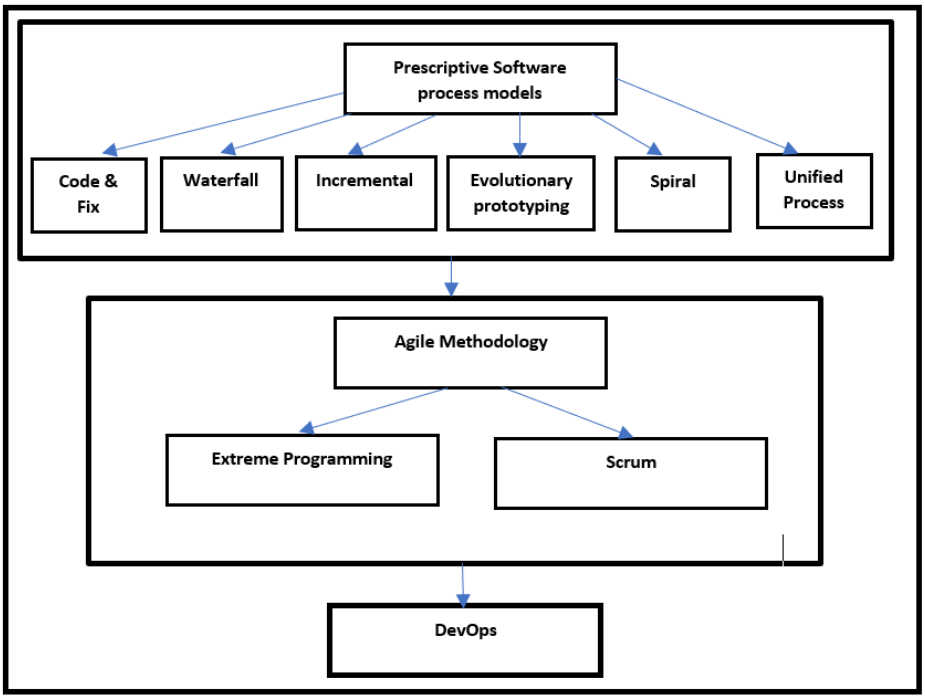

Figure 2.2: Historical perspective on software development (Own Work)

2.2. EARLIEST SOFTWARE DEVELOPMENT USING CODE AND FIX

The earliest software development process introduced in the 1940s to 1960s was characterised by a “Code and Fix” method (Boehm, 1988). In this method, a developer approaches software development as a two-step process; one step is to write some code, and the other step is to fix it based on the output (Figure 2.3). They are also referred to as “Cowboy coding”, and this process is carried out iteratively until the programmer is satisfied that the intended requirements of the software are met. According to Brooks (1987), the main objective of the programmer is to create a software artefact instantly by focusing on writing code. This process is not feasible for software with significant functionality since it is only appropriate for 100-200 lines of coding.

Due to its unstructured nature, this process resulted in software not meeting user requirements and being challenging to maintain due to a lack of documentation. Boehm (1988) highlights three main difficulties with this model:

- Due to the number of fixes, the code lacks proper structure, making subsequent fixes very costly.
- Even when the software was frequently well-designed, it did not align with the users' needs. It was therefore rejected by users or resulted in the costly task of redeveloping the software.
- Due to a lack of testing and maintenance activities, code became costly to fix and maintain.

The challenges demonstrated a need for a more structured approach to software development, referred to as the software process.

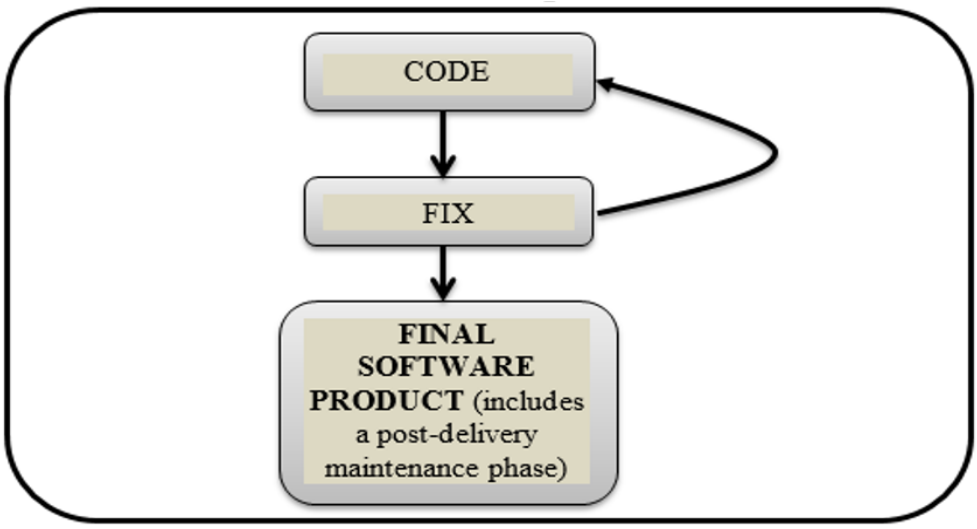

Figure 2.3: Code and Fix software development process (adapted from (Schach, 2008))

2.3. THE SOFTWARE CRISIS AND SOFTWARE PROCESS

According to Fitzgerald (2012), the term software was coined in 1958. Within ten years of conceptualisation, Naur and Randell (1969) used the term "software crisis" to highlight software development and implementation difficulties. These difficulties resulted in software taking longer to develop, costing more than initially budgeted and not working as anticipated. To address this problem, a group of experts in 1968, during a NATO Software engineering conference, put forward the notion that software development should follow similar principles as traditional engineering disciplines, which led to the domain of software engineering. This led to a more structured approach to software development that resulted in the creation of software processes and software development methodologies (Naur, 1968).

Pressman (2010) refers to the software process as activities, actions, and tasks that must be completed in order to create software. Sommerville (2007) also reiterates that the software process is a disciplined approach used when software is engineered by using a sequence of activities leading to the development and implementation of a software product. Software specification, development, validation, and evolution are inherent to all software processes. Schach (2008) further states that the software process is how software is produced, which incorporates the methodology, software life-cycle model, techniques, tools and the individuals building the software. Pressman (2010) also states similar activities to Sommerville (2007) resulting in the conceptualising of these activities into a generic process framework. These activities are communication, planning, modelling, construction, and deployment. A

simplified representation of a software process is referred to as a software process model since only partial information about a software process is depicted in a software process model (SPM) because it represents a process from a particular standpoint. Schach (2008) refers to these process models as life cycle models, which describe the steps that should be performed when building a software product. Pressman (2010) further describes each model as having a process flow that interconnects interrelated process elements. Different process models are distinguished from each other based on the varying emphasis on framework activities and how the process flow invokes various framework activities differently.

The SPM provides the foundation for creating software development methodologies (SDM). The SPM illustrates "what needs to be done", while SDM demonstrates "how to do it". The "what to be done" in SPM is achieved by the SPM providing a "bird's eye view" of the sequences that need to be followed when transitioning from one software development phase to another, as well as the milestones required to be met in a certain step that will allow for the advancement to the next step. SDM provides an operational perspective of the development process by providing details of the software development methods that the software development teams must carry out. These methods include requirement gathering techniques, analysis and design tools and techniques that enable the team to achieve their goal for each of the phases in the SPM (Pressman, 2010; Sommerville, 2007).

SPM can be categorised as either plan-driven or agile processes. Plan-driven processes have all activities planned in advance, and software development progress is measured against this plan. Pressman (2010) also coins the terminology prescriptive process models to describe plan-driven SPM. Prescriptive process models emphasise detailed definition, identification, and application of process activities and tasks. Several studies (for example Awad (2005); Kent et al. (2001); Khan et al. (2012)) use the term "heavyweight process models". They intend to improve system quality, make projects more manageable, make delivery dates and costs more predictable, and guide teams of software engineers as they perform the work required to develop software. Unlike plan-driven processes, agile processes are planned in increments and can adapt more quickly to changing customer requirements. Boehm and Turner (2003) discussed that each approach is suitable for different types of software. Plan-driven approaches are suitable for mission-critical software; if requirements in the system are incomplete and there is a lack of quality, it can result in the loss of human life or injuries. However, agile processes are more suited to business environments that are seeking operational efficiency and competitive advantage.

2.4. PRESCRIPTIVE PROCESS MODELS

2.4.1. Waterfall Model

Bennington, in 1956, proposed a nine-step sequential, procedural model used by the United States (US) Air Force to assist the development of software to control air defence systems (Benington, 1983). In 1970 it was modified by Royce into a seven-step model consisting of system requirements, software requirements, analysis, program design, coding, testing, and operations phrases. The term 'waterfall' was not coined by Royce but first appeared in a paper titled "Software Requirements: Are They Really A Problem?" (Bell & Thayer, 1976). Since this model focuses on the definition and analysis of requirements before any design and coding, it created a strong foundation for models that were subsequently created (Ruparelia, 2010).

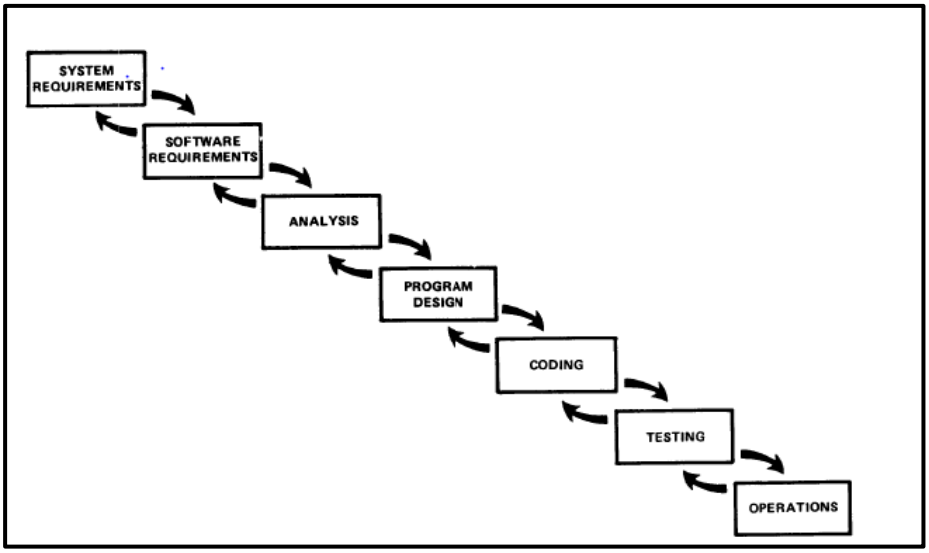

Figure 2.4: Waterfall Model stages (Royce, 1970)

Figure 2.4 illustrates the interplay of the different phases of the life cycle model. Below is a brief description of these phases as highlighted by (Royce, 1970).

System Requirements: During this phase, the hardware requirements and the software tools necessary for building the new system are identified.

Software requirements: Establish the broad software functionality of the new system.

Analysis: These general requirements are investigated in terms of business processes and business functions to create the requirement specification document. This phase establishes what the newly developed system is going to accomplish.

Program Design: Using the requirement specification document, the software is designed to illustrate the system's requirements. This phase establishes how the conditions will be incorporated into the new system.

Coding: Using a high-level programming language, the program design is implemented.

Testing: Various tests such as alpha, beta, integration and user acceptance tests are carried out during this phase.

Operations: During this phase, the system is implemented into the operational environment.

These seven steps have been refined and renamed over a period by different organisations and researchers, with Pressman (2010) documenting the phases as communication, planning, modelling, construction and deployment. Sommerville (2007) refers to the water phases as requirements definition, system and software design, implementation and unit testing and operations and maintenance, with Schach (2008) calling these phases the requirement, analysis, design, and implementation phases, postdelivery maintenance and retirement. Irrespective of the changes to the number and names of the waterfall phases, all waterfall models are underpinned by the structured approach, explained by Royce (1970).

The waterfall model has various strengths, making it a popular choice within software development organisations. With each phase distinct from the other, with its own deliverable and documentation, it is easy to understand and implement, enabling managers to monitor progress against a development plan (Sommerville, 2007). Tipaldi et al. (2013) further state that the waterfall model promotes the testing and validation of each artefact and outcome. Waterfall also promotes good habits that force software developers to define and document requirements before design and detailed design before coding. Despite these benefits, there are various criticisms against this model. Kess and Haapasalo (2002) indicate that many real-life software projects do not follow a linear flow as the waterfall model suggests since gathering all requirements without iterations of phases is challenging.

Hence, the time required before a project can be marketed or ready for use by the client is long, preventing the client from gaining early benefits from the software. Boehm (1988) argues that the biggest problem in the waterfall model is the heavy documentation in the requirements definition and design phases. Schwaber (1995) points out that the incapability of the waterfall model to react to changes in the middle of the linear process causes considerable problems. Furthermore, if requirements are missed or not captured according to the client, it becomes too expensive to fix later in the development cycle (Patel & Jain, 2013).

2.4.2. Incremental Model

Schach (2008) highlights two aspects affecting real-world software development using the waterfall model. One aspect is the moving target problem, where software requirements constantly change while the software is being developed. The second is known as Miller's Law, named after George Miller, a Professor of Psychology. He states that humans cannot concentrate on more than seven chunks of information at a time (Miller, 1956). In most instances' software being developed typically has much more than seven requirements. With the gathering of all requirements before design and coding can begin, it will be difficult in the waterfall model for developers to concentrate on all requirements during a single pass of the cycle, with software implementation occurring as one 'big bang' in the end. Therefore, building software using stepwise refinement is necessary to address this aspect. Stepwise refinement lets developers focus on the essential functionality first and postpone less critical functionality to the next increment. According to Larman and Basili (2003), iteration and incrementation are intrinsic components of software engineering used in conjunction to create a software artefact. Incrementation is when a software artefact is constructed chunk by chunk, and iteration is when each chunk or increment has to go through several versions.

Basili and Turner (1975, p. 395) mention that software process models that incorporate both Iterative and Incremental development (IID) methodologies would require a "…sequence of successive design and implementation steps, beginning with an initial 'guess' design and implementation of a skeletal sub-problem."

An example of this model is called the Incremental software model. According to Pressman (2010), the Incremental model is used when initial software requirements are well understood, but a purely linear process cannot be followed for various reasons. Instances of these are when there is an urgent need for the client to have a limited set of functionalities quickly. This results in applying the linear sequences of the waterfall model in a staggered fashion as calendar time passes. The initial increment delivered to the client is the core product.

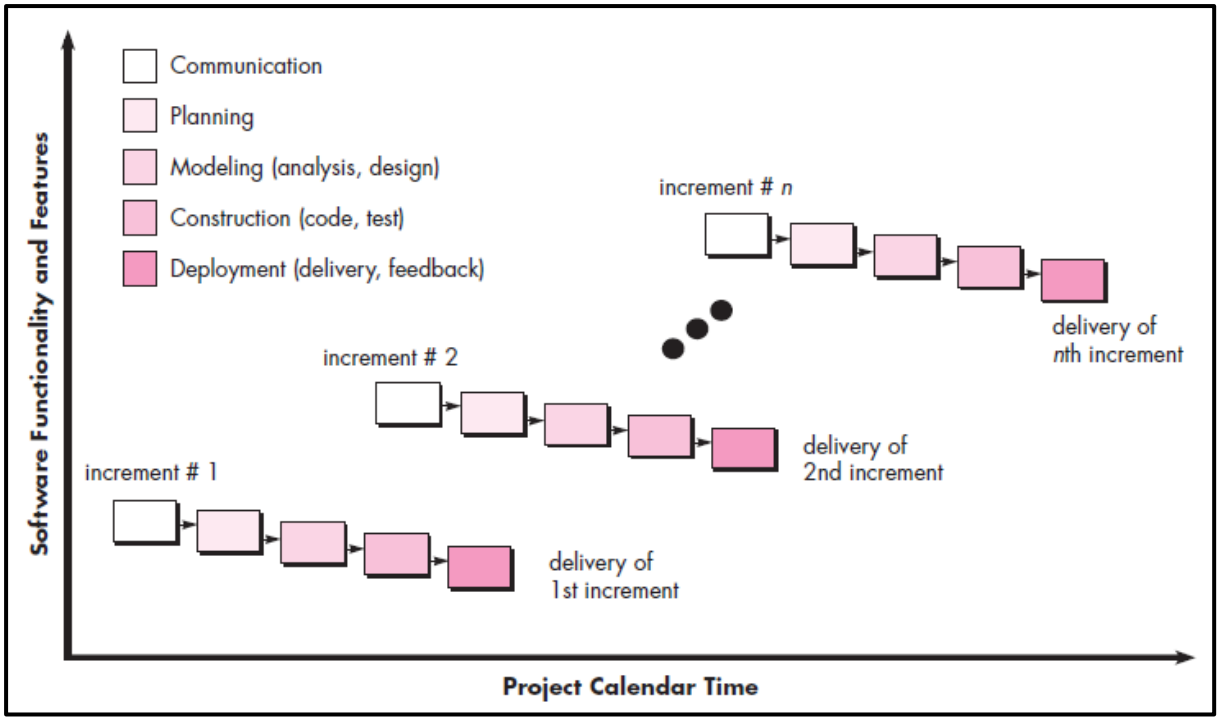

Figure 2.5: Incremental model (Pressman, 2010)

As project calendar time progresses, the software evolves into the final software product. This is achieved by eliciting constant user feedback with the delivery of each iteration, allowing for the refinement of the system's design and evolution to a final software product solving the actual problem and increasing the maintainability and robustness of the final product (Pressman, 2010). The ability to incorporate client feedback from previous iterations in the recent iteration is one of the significant strengths of the Incremental model. This ensures that the final system will meet the client's requirements. Further strengths are the early identification of architectural risks, allowing clients to obtain early benefits from the system, and risk mitigation by identifying and resolving issues early (Ruparelia, 2010). Similar to the waterfall model, the disadvantage of this model is that it requires the creation of intensive documentation. Furthermore, due to new software components being incrementally integrated with previously created components, the overall system architecture must be established in advance; otherwise, expensive system architecture or design issues can result (Bhuvaneswari & Prabaharan, 2013).

2.4.3. Evolutionary Models

According to Pressman (2010), software is complex, resulting in software requirements not being fully understood at the beginning of the software development process. This software needs to evolve over a period of time. Sommerville (2007) states that changes are predictable

during the development of large software systems. These modifications result from constant changes in the business environment, which leads to new requirements that need to be implemented in the system under development. Furthermore, due to companies constantly enhancing their business processes to attain efficiency and effectiveness to become innovative and obtain competitive advantages, software requirements often change as development proceeds. This makes it unrealistic to follow the straight path of the waterfall model and, therefore, necessary to divide the project into increments, as alluded to in the incremental model. In this instance, as developers understand the requirements, the software products evolve into a final product with core requirements developed to address business and competitive pressure. The evolutionary prototyping and spiral models are two dominant models created for software development that evolve (Pressman, 2010; Schach, 2008; Sommerville, 2007).

Evolutionary prototyping

Prototyping can form part of the model requirement gathering phase or be used to create a throw-away prototype as proof of concept to gather requirements or an evolutionary process model (Davis, 1992). Prototyping is the process of creating an initial version of the software to assist in the understanding of user requirements. The artefact created is a prototype with which the users interrogate and provide feedback to developers on what changes can be made (Floyd, 1984). Prototypes often undergo various revisions until users and developers agree that the prototypes capture the users' required functionality. At this point, the prototype can be thrown away, and the software development process continues with the development of the software based on the understanding of the functionality obtained from the prototype, or the prototype can be developed into the final software. The process where a prototype evolves into a final software product is referred to as evolutionary prototyping, and the model used to achieve this is the evolutionary process model (Carr & Verner, 1997).

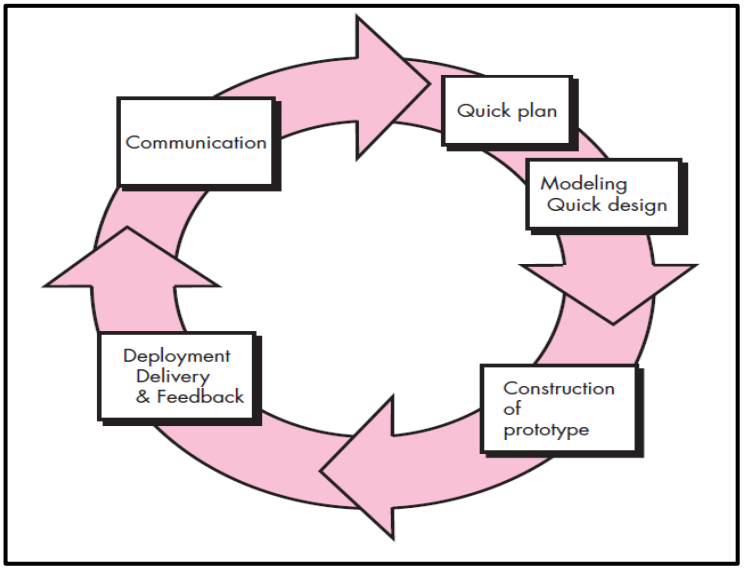

Figure 2.6: Prototyping model (Pressman, 2010)

The phases in the model are communication, a quick plan, quick modelling design, prototype construction, deployment, delivery and feedback. During the first iteration of the communication phase, general requirements of the systems is communicated to the development team. Using these requirements, a quick plan of what features the system may have, is determined, followed by modelling and fast design. After that, a prototype based on the design is created and provided to the client for feedback. This results in the communication of new changes to the prototype, which results in another iteration. The iteration process will continue until the client is satisfied that the software product meets their requirements. The software product's functionality is enhanced with each iteration, resulting in incremental development. After the client approves the functionality, non-functional requirements are added to the software before it is delivered and deployed. (Pressman, 2010; Schach, 2008; Sommerville, 2007). The advantages of a Prototyping approach are that it enables the development process to be broken down into more manageable steps (Kraushaar & Shirland, 1985) and is cost-effective because client feedback is constantly sought to prevent the cost of rework (Boehm et al., 1984; Gordon & Bieman, 1995; Palvia & Nosek, 1992). Furthermore, greater collaboration and communication between the development team and the end users (Alavi, 1985, Naumann and Jenkins, 1982) help determine the technical feasibility of client requirements early, thus reducing risk (Alavi, 1984; Floyd, 1984; Naumann & Jenkins, 1982; Tate, 1990). Kimmond (1995) highlights various disadvantages of the evolutionary prototyping process, indicating that there is reduced project management control due to the constant changes that the end-user may prescribe. It further creates a false sense of user expectation, with the user anticipating every suggestion to be incorporated into the system. Due

to the constant changes made in the system it is also difficult to properly document the system being developed. Chandra (2015) also alludes to the model being time-consuming due to the constant development and feedback loop between developers and end-users, which will also result in the project costing more.

Spiral Model

The spiral model, which integrates the iterative properties of prototyping with the structured aspects of the waterfall model, was proposed by Barry Boehm (Boehm, 1988). This model aimed to provide rapid software development, considering the risks that may exist in a project. The model is represented by a spiral, with each loop in a spiral representing a phase in the software development process. Boehm and Hansen (2000) state that its cyclic approach that grows software incrementally with its concurrent decrease of risk and a set of milestones that need to be achieved as development progresses are the two distinguishing features of the spiral model. Furthermore, the spiral model incorporates the best practices of both the waterfall and prototyping models. (Mujumdar et al., 2012; Munassar & Govardhan, 2010). The development team starts at the centre of the spiral with a few requirements. It then goes through the development phases, enabling the developers to better understand the needs and evaluate the risks. As the development stages progress to the outer spiral, more requirements are added to the development until the software has met its functionality and quality to be installed and maintained (Alshamrani & Bahattab, 2015).

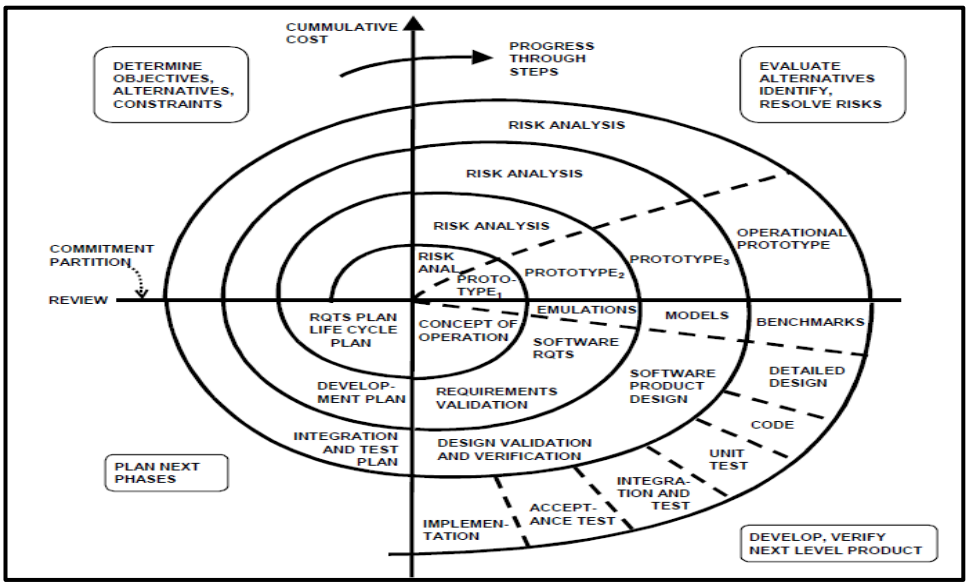

Figure 2.7: Spiral Model (Boehm, 1988)

As illustrated in figure 2.7, the Spiral model is divided into four quadrants, each comprising a phase in the lifecycle. The first quadrant is the objective setting that identifies explicit objectives for the project. The second phase is the risk assessment and reduction phase, where significant risks are identified and analysed to reduce the risks. During the third quadrant, development and validation of the product are undertaken. During this phase, requirement validation and detailed design are carried out. Also, coding occurs together with different types of testing. During the fourth quadrant (reviewing and planning), the product created so far is reviewed with the customer, and the next iteration of the spiral is planned. With each iteration around the spiral, a complete version of the software is gradually built (Boehm, 1988).

The advantage of the spiral model is a dedicated phase for evaluating risks that ensures that projects with a high risk of failure are reviewed during each spiral. Therefore, the spiral model is suitable for technically challenging projects. Furthermore, it provides an opportunity for the end-user to interact with the prototypes increasing the likelihood of success. The weakness of this model is that a significant amount of time can be wasted on the risk analysis phase for low-risk projects. Since it is a highly complex model, you need well-trained personnel in risk analysis to identify and mitigate risks (Alshamrani & Bahattab, 2015; Boehm, 1988; Chandra, 2015; Ruparelia, 2010).

2.4.4 Rational Unified Process

With the increased use of object-oriented programming to develop software, Rumbaugh, Booch, and Jacobson wanted to create uniform methods to document object-oriented design (Jacobson et al., 1999a). This resulted in the development of the Unified Modelling Language (UML), which became the "de facto industry standard for object-oriented software development" (Pressman, 2010, p. 54). However, as much as this provided the technology to assist in object-oriented software engineering, there was still a lack of a process model that helped software development teams navigate the application of object-oriented technologies. This led to the development of the unified process, a framework that uses UML to develop object-oriented applications (Jacobson et al., 1999a).

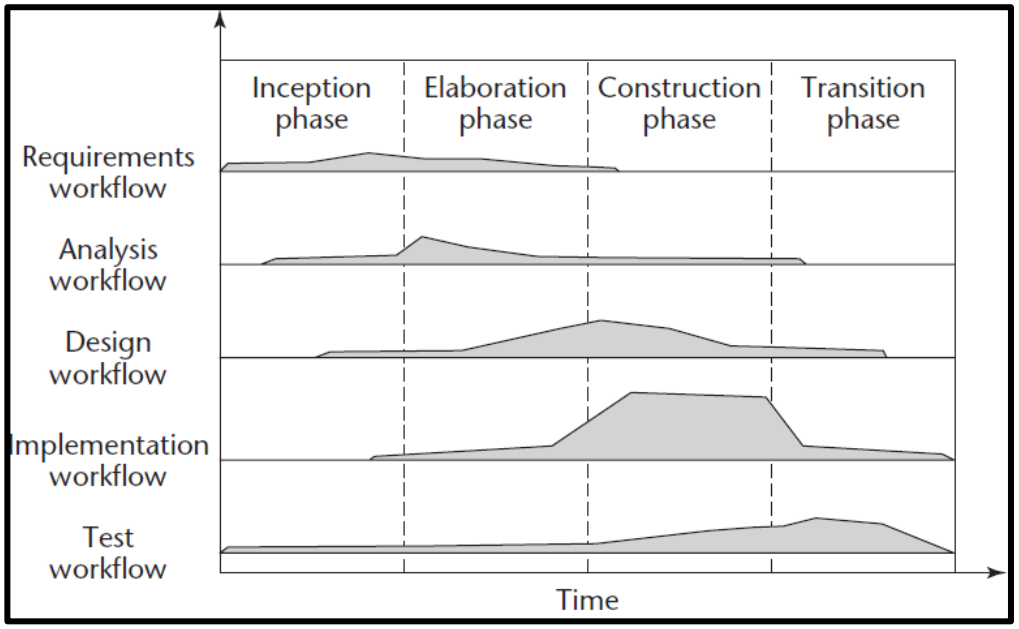

Figure 2.8: The Unified Process (Schach, 2008)

As illustrated in Figure 2.8, the Unified Process (UP) is a two-dimensional model consisting of workflow activities and phases. This two-dimensional architecture differs from previous models that incorporated workflow activities as distinctive phases within a software model and, therefore, could not continue workflow activities in other parts of the life cycle. The two-dimensional nature of UP portrays the more dynamic and realistic approach to software development in that the same workflow activities may need to occur in different parts of the software development lifecycle (Schach, 2008).

The Inception Phase mainly focuses on determining the system's objectives and the economic feasibility of developing the system. Preliminary use cases are designed to describe users' primary functionality and features. Furthermore, resources are identified, significant risks are assessed, and a schedule is defined. The Elaboration Phase entails refining and expanding the preliminary use cases created in the inception phase. These models are represented using the UML. In the Construction Phase, coding and testing activities are used to convert the use cases established in the elaboration phase into software components. Software components created in previous iterations are integrated with the newly developed software components. The Transition Phase entails an operational version of the software being handed over to end-users for feedback and beta testing. The software team also creates documentation such as user manuals, troubleshooting guides, etc. The software increment may be released into the

operational environment depending on the number of iterations completed (Jacobson et al., 1999b; Pressman, 2010; Schach, 2008; Sommerville, 2007).

According to Ambler (2005), UP provides various advantages such as the continuous early deployment of software and frequent interaction with stakeholders, thereby ensuring that delivered software components align with the user's requirements, allowing for higher risk requirements to be handled earlier, thus improving risk management by focusing on the creating software in small increments allows for the project to adapt to changes. It enables developers to concentrate on delivering working software to clients as soon as possible, thereby improving IT productivity. However, Khan et al. (2012) cite various weaknesses in UP, such as complexity that necessitates the hiring of experts, lack of open and accessible standards due to it being a commercial model owned by IBM and its unsuitability for small software development organisations due to its cost.

2.5. AGILE SOFTWARE PROCESSES

The processes discussed previously are considered heavy-weight processes since they follow a plan-driven approach that focuses heavily on design activities and documentation. As a result, these processes cannot easily accommodate changes to software requirements during the software engineering process. According to Pressman (2010), this ability to adapt to changes, especially modernised web-based economy where it is difficult to determine how software systems will transform over time. Fowler (2005) further states that plan-driven processes are frequently critiqued as bureaucratic due to the many phases and activities that must be followed, resulting in more extended development. Sommerville (2007) asserts that using the plan-driven approach has too much overhead of planning, designing, and documenting, making the process not feasible for business systems where software requirements frequently change, resulting in a rework of the product in terms of specification, design, and programming. Cohen et al. (2004, p. 3) also recognised the "need for an alternative to documentation-driven, heavyweight software-driven processes." The agile methodology was thus put forward to address the plan-driven approach's shortcomings. The agile approach enabled software developers to concentrate more on software development activities than design and documentation tasks (Dwivedula & Bolloju, 2020; Sommerville, 2007).

Furthermore, agile methods provide the capability to handle changing requirements (Cockburn, 2002). Leffingwell (2010, p. 16) also states the following regarding Agile Software Requirements, "We take a far more flexible approach to requirements management: one that is

temporal, interactive, and just-in-time." This statement is enforced by Figure 2.9, which illustrates estimated or flexible requirements in the agile approach with resources and timelines fixed. By comparison, requirements are fixed in the traditional approach, and resources and timelines vary.

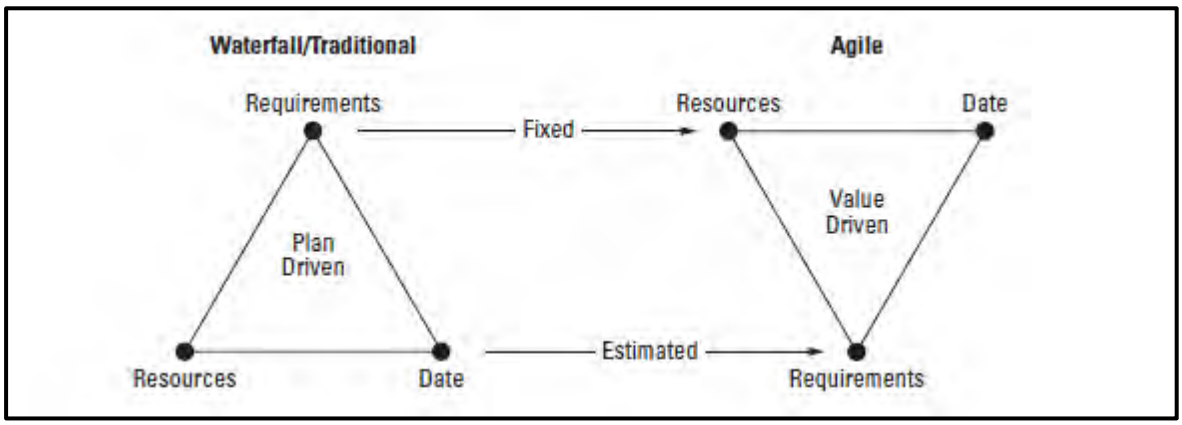

Figure 2.9: Agile Iron Triangle (Leffingwell, 2010)

2.5.1 Agile Manifesto and Agile Principles

A set of values and principles underpins the agile approach. The values are enshrined in the agile manifesto, which was established to diverge from traditional practices that focused predominantly on tools and techniques, intensive documentation, elaborate planning and contract negotiation to interaction and collaboration, and delivering working software early customer collaboration and responding to change. These values are classified as Individuals and interactions over processes and tools, collaboration of customer over contract negotiation, working software over comprehensive documentation, and responding to change over following a plan (Kent et al., 2001).

Individuals and Interactions over processes and tools

Processes and tools are essential for the success of a development project, but more important are the interactions between team members and the customer to understand what is required in terms of the software's functionality. In many instances, processes and tools are the main focus, with no understanding of the system's requirements leading to software failure. Hence, it is vital for an individual to interact with the team and the client to understand the client's needs (Beck et al., 2001; Fowler & Highsmith, 2001; Hohl et al., 2018; Zaitsev et al., 2018)

Delivering working software over comprehensive documentation

Although documenting requirements, analysis, and design changes are essential during a project, delivering working software is more important to a client. Therefore, the agile manifesto places more value on the development activities of the software development lifecycle that will result in giving the client early delivery of software, enabling the client to obtain the most significant benefit of the software product as soon as possible (Beck et al., 2001; Fowler & Highsmith, 2001; Hohl et al., 2018; Zaitsev et al., 2018).

Collaboration with customer over contract negotiation

Most software development divisions or organisations want to negotiate with the customer the contract details, such as the system's requirements, before they begin the development effort. It will allow them to adequately scope, budget, and allocate resources to the project. However, the agile mindset is to collaborate with the customer to enable the changes to the requirements helping to achieve the customer's immediate needs and also creating a product that will meet the customer's operational environment (Beck et al., 2001; Fowler & Highsmith, 2001; Hohl et al., 2018; Zaitsev et al., 2018).

Responding to change than following a plan

Creating a plan that provides timing and resource allocation details is fundamental to all software development projects. However, the plans created using traditional process models have too many details that become inflexible to change. Therefore, the agile methodology advocates for plans that can respond to change (Beck et al., 2001; Fowler & Highsmith, 2001; Hohl et al., 2018; Zaitsev et al., 2018).

According to Fowler and Highsmith (2001), the agile manifesto set of values is underpinned by the 12 principles. Palopak and Huang (2024) identified a substantial correlation between adopting agile principles and project performance, with regular delivery, technical excellence, team member proactivity, and customer collaboration exhibiting the most significant positive impact on Agile project success. The agile principles are as follows:

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software. Unlike the waterfall approach, where the client waits for an extended period to use the software since all requirements are delivered at once during the end of the lifecycle, agile provides software continuously to the client.

- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage. Hence customers can add requirements that they feel will enhance the efficiency of the operations or give them a competitive edge while the project is being developed.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference for the shorter timescale; in contrast with the waterfall approach, the software is delivered and implemented in one big bang at the end of the project.
- Business people and developers must work together daily throughout the project. The project team must consist of both business people and end-users who interact with each other throughout the entire project so that requirements and implementation decisions can be discussed and clarified immediately. There is no misunderstanding of the requirements that may delay implementation.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to do the job. Team members working in an agile environment must have faith in the method and be motivated to use the principles and philosophy of agile approaches to get the project done. Team leaders and members must trust that each other has the skills and knowledge to achieve the project deliverables.
- Face-to-face conversation is the most efficient and effective method of conveying information to and within a development team. In agile, the timeframe is fixed, so any unnecessary delays in the project must be prevented. To overcome this, face-to-face communication is essential so the discussion can be done immediately, allowing for efficient and effective decision-making and preventing delays in the project.
- Working software is the primary measure of progress. The traditional approach creates multiple documents for requirements, analyses, and design decisions. These documents can create a false impression of progress towards the ultimate goal of a working piece of software. In agile software development, progress is only made when working software is developed and given to the client.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely. Agile teams need to work constantly daily for agile projects to be successful. Any period where development does not occur will result in a delay in meeting that deliverable. Therefore, the agile approach does not advocate overtime and overtime work because if team members can sustain daily performance, the project milestones will be attained.

- Continuous attention to technical excellence and good design enhances agility. Due to the continuous delivery of software increments, they must be designed and implemented with the highest quality, ensuring they are seamlessly integrated with previously developed increments.
- Simplicity - the art of maximising the amount of work not done is essential. Often, unnecessary models and documentation are created during the traditional approach that does not add value to the software development process. In agile, the philosophy is to carry out tasks that will benefit and add value to the value software project.
- The best architectures, requirements, and designs emerge from self-organising teams. In agile development, the project manager does not allocate tasks to individuals. The development teams assess their strengths and weaknesses and decide which tasks team members are assigned. This will harness individuals' strengths, resulting in more successful software.
- The team regularly reflects on how to become more effective, then tunes and adjusts its behaviour accordingly. At the end of every increment, the team members must reflect on the positives and negatives that occurred during the increment. This will allow them to take the positives and eliminate or change the negatives into positives.

2.5.2 Characteristics of Agile Development

Modularity and Iterative

Modularity in agile methodology is achieved by breaking software requirements into fragments, with these different fragments being developed over various iterations. During each iteration, a deliverable is developed, which is then enhanced in subsequent iterations until all software requirements are met (Abbas et al., 2008; Avison & Fitzgerald, 2006; Sajannavar & Dharwad, 2022).

Adaptive

Agile methodology can adapt to changes and risks by re-planning activities and effecting changes to the software development activities if necessary. Stakeholders welcome change during the project lifecycle and view changes as necessary to ensure that the correct requirements are captured so that the project will meet the client's needs and succeed (Abbas et al., 2008; Fowler, 2005; Niko, 2007; Sharma et al., 2012).

Incremental

Agile methodology focuses on releasing software into the operational environment in increments. Unlike the waterfall model, which releases software as one big bang, in the end, this incremental release will allow the organisation to gain a competitive advantage and benefits from the software as early as possible. As more increments are developed, they are integrated into the existing operational increment (Abbas et al., 2008) .

People-oriented

Agile methodology's success depends on the team members' competence rather than the processes and technologies used in agile (Yong, 2021). Since risks cannot be identified upfront in an agile project, team members will need the experience and knowledge to identify and mitigate them as the project progresses (Avison & Fitzgerald, 2006). Cockburn and Highsmith (2001, p. 131) also state, "The most important implication to managers working in an agile manner is that it emphasises people factors in the project: amicability, talent, skill, and communication".

Collaborative

To succeed in agile methodology, constant communication and collaboration between agile team stakeholders must be needed. Since agile projects do not do detailed upfront planning, adapting to changing system requirements is necessary. Therefore, agile members must properly collaborate to plan future increments and discuss changing user requirements (Avison & Fitzgerald, 2006).

2.5.3 Traits of Agile Team Members

Pressman (2010) and Zainal et al. (2020) state that team members must have some vital traits to successfully apply the agile process's characteristics.

Competence

Team members within the agile environment must have the necessary software engineering and agile development skills and knowledge (Pressman, 2010). Cockburn and Highsmith (2001) also believe that if you have competent people, no matter what process is followed, the project is likely to succeed. This is highlighted in the agile manifesto that favours "individuals over processes and tools".

Common focus

All team members must have the same focus, delivering working software to the client as soon as possible. Even though all members might play different roles in the agile team and have different goals and tasks, they must do so with a common focus when performing their tasks (Pressman, 2010). Waldron (2017) highlights how having a shared goal with a team, results in greater collaboration, enhancing the success of agile development practices within an organisation.

Decision-making ability

Time is significant in agile development. Therefore, to prevent unnecessary waste of time, the agile team must have decision-making abilities regarding technical and project issues. This will allow individuals and the team to make decisions without waiting for someone in higher authority to dictate decisions (Drury et al., 2012; Pressman, 2010).

Fuzzy problem-solving ability

The agile team will encounter many unclear and unstructured problems during development. The team must have the ability to handle these. Furthermore, the difficulties faced today will differ from the ones they will encounter tomorrow due to the unique nature of different software development projects. Therefore, the solutions to problems will be different. The teams must be able to deal with these ambiguities and change continually (Strode et al., 2022).

Mutual trust and respect

Since each team member will be responsible for achieving specific tasks, they must have faith and respect for their team members in getting the work done. DeMarco and Lister (2001)use the words "jelled team" to indicate teams that trust and respect each other. Crittenden (2016) further state that "Jelled teams are made up of people who are motivated (because they are working on things that give them Purpose, Autonomy, and Mastery), and they find happiness in working together. With a group like that, it's hard not to succeed".

Self-organisation

In an agile environment, the team organises themselves for the product to be created. This means that a team leader or project manager does not allocate work; rather, it entails the team identifying the team members' strengths and assigning tasks to members that can best achieve this. Schwaber (2010, p. 1) states that a self-organising team "selects how much work it believes it can perform within the iteration, and the team commits to the work. Nothing

demotivates a team as much as someone else making commitments for it. Nothing motivates a team as much as accepting the responsibility for fulfilling commitments that it made itself." Furthermore, Stettina and Heijstek (2011, p. 94) state that team autonomy is critical and a "lack of team autonomy is believed to lead to excessive overtime, high defect rates and personnel burnout".

There are many agile methodologies, Extreme Programming, Scrum, DSDM, Feature-Driven, etc.). A complete comparison of primary methodologies is beyond the scope of this thesis. Therefore, the two principal methodologies used in industry are discussed, i.e. Scrum and Extreme Programming (Al-Zewairi et al., 2017). Discussing these two methodologies will explain how agile principles and practices are incorporated into a software process.

2.5.4 Extreme Programming

Extreme programming (XP) is an agile software development methodology that advocates frequent "releases" in a short development cycle, enabling improved software product quality by being more responsive to changing customer requirements. They are closely associated with the values and principles of the Agile Manifesto, Beck (1999) and Beck (2000) document XP's values, principles, and practices.

XP values

Communication, simplicity, feedback, respect, and courage are the core values of XP. Team members need constant communication during every project phase, reducing the possibility of misunderstanding or misinterpreting the project requirements. Developers must follow the 'keep it simple (KIS)' philosophy by creating simple designs and code, reducing time and effort. Feedback is obtained from frequently delivered software, thereby increasing the quality of the software product. For the project to be successful, there needs to be an understanding that all team members are working towards a common goal and that mutual respect is needed among team members. Developers are encouraged to evaluate their work to find errors and areas of improvement and take constructive criticism by allowing others to assess their work (Beck, 1999; Beck, 2000; Sambasivam, 2004).

Core principles and practices of XP

Kuppuswami et al. (2003) illustrated that software projects implementing XP practices decreased development effort. This was further substantiated by Kobayashi et al. (2006), who found an increase in communication, productivity, maintenance, and code quality and

requirements in projects that implemented XP principles and practices. Mannaro et al. (2004) compared the job satisfaction of developers that use XP practices with those that do not. Developers using XP practices were more satisfied since XP practices created a more conducive development environment, resulting in increased productivity. The XP practices are tabulated in Table 2.1.

Table 2. 1: Core Principles of XP (Adapted from (Beck, 1999))

Coding standards	Since many programmers are working on a project, each having their own individual programming style, coding standards must be established before a project begins. This will ensure consistency within the code, including the maintainability and testability of the software.
Simple Design	Design within XP advocates a "design with a purpose" principle, meaning that designs are created to understand the functionality that will be developed. Therefore designs are kept to a minimum. There should be no unnecessary design steps undertaken. This Simple Design: YAGNI stands for 'You Aren't Gonna Need It', which suggests that leave it out if something is not required.
Small releases	For the customer to benefit from the software as soon as possible, the software can be released to the customer after each iteration. The software can be installed into the operational environment, or the customer can test the release. Both will provide valuable feedback to the developers allowing them to improve on the release.
Collective code ownership	Every developer is responsible for all lines of code. Developers share ideas and allow their code to be inspected, allowing developers to interact and collaborate.
Planning game	User stories are used to document the most valuable features that customers want to be implemented. User

	stories contain descriptions of the scope, the number of days to develop and the release date for the features.
Metaphor	System metaphor removes technical jargon when expressing user stories allowing for a simple design. Members in the team will thus understand this simple design. This will enable developers to immediately begin work on the user stories without trying to decipher specifications (Grinyer, 2007).
Simple Design and refactoring	There is no detailed upfront design of the system, but rather designs are simple and focused on individual user stories. Furthermore, code is optimised by refactoring, a process whereby the internal structure of code is improved without changing its behaviour.
Tests-Driven Development	XP is known as a test-driven methodology (TDD). A set of test cases are compiled by programmers and customers when user stories are documented. These test cases are used to test features completed during the coding phase,
Pair Programming	All programming code is written by pairs of developers working on a single computer.
On-site customer	A customer is part of the development team working with developers daily.
Continuous integration	Within a short time frame, new code is continually merged with the previous release of the system.

XP Process

The XP values and practices are incorporated into the XP process. As illustrated in Figure 10, this process consists of six phases; the exploration, planning, iteration to release, the productionising phase, maintenance, and death (Abrahamsson et al., 2017).

The **exploration phase** is essential to the success of the entire XP process because it entails gathering the initial requirements for the software. These requirements are expressed as user

stories by the customer. The development team also determines technical feasibility, familiarising themselves with the tools and technology used in the project. An initial prototype is developed to understand the requirements and determine the initial architecture of the system. During the **planning phase**, the development team assesses each user story to determine which will be the mandatory and optional requirements and determine the length of time to deliver each user story. After that, a group of user stories is grouped into increments, and delivery dates are determined. In the **iteration and release phase**, the software increments identified during the planning phase undergo several iterations through the design, coding and testing activities until they are ready for the production system. Each iteration lasts between one to four weeks. The "KIS" principle is followed during the design phase, whereby simple designs are created. The coding phase entails developers working in a paired programming paradigm and the code undergoes continuous refactoring to improve its internal code structure. During this phase, unit tests are also executed. After the completion of each increment, project velocity is used to determine the timeframe for the subsequent sprint. This is achieved by examining if all the user stories in the first sprint were achieved within the time allocated for the sprint. If all was not completed, fewer user stories were added to the subsequent sprint; if all was accomplished with time remaining in the sprint, more were added to the next sprints. During the **productionising phase**, increments are released into the operational environments. Before it is implemented, the system performance is tested again. This phase enables the integration of new increments with existing increments. The **maintenance phase** develops any requirements or new ideas not implemented previously are implemented during this phase. The **death phase** can result from one of two processes. In the first process, if all user stories are completed, and the customer's requirements are met by the system or in the second process, a decision can be made to terminate the project due to its high cost or not having the desired effect on the organisation.

For the above to be successful, there needs to be constant interaction between multiple stakeholders within the XP process. The key roles are the programmers, customers, testers, and managers.

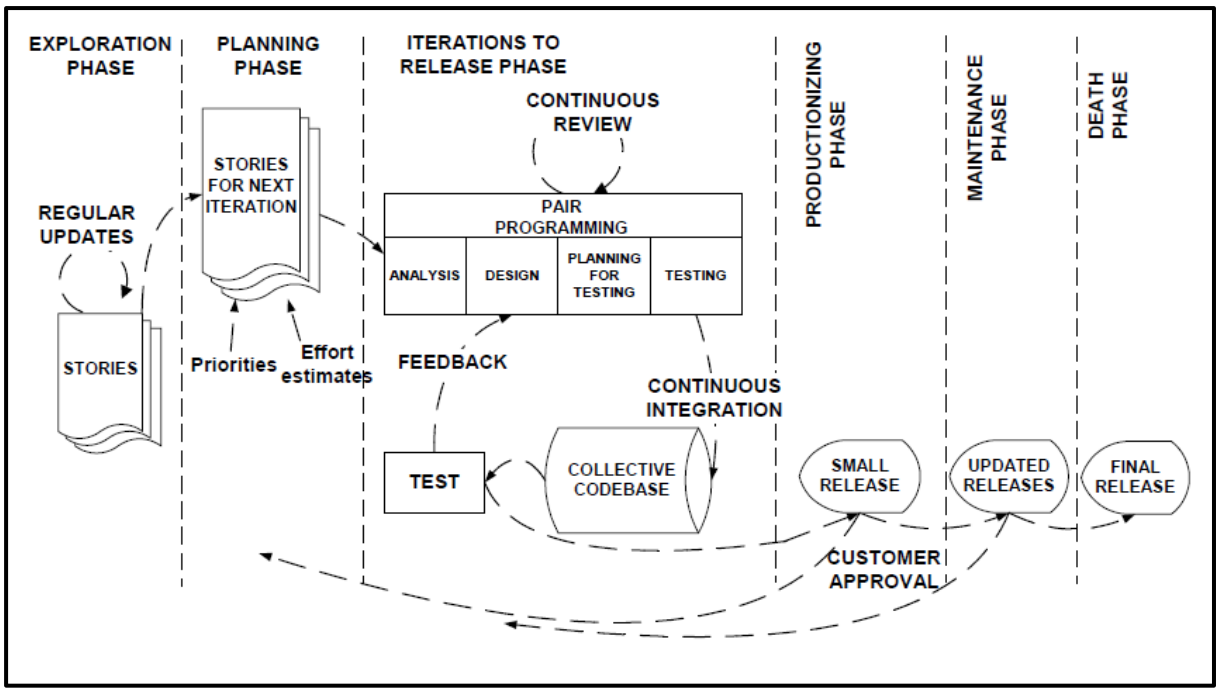

Figure 2.10: XP process model (Abrahamsson et al., 2017)

According to Abrahamsson et al. (2017), the advantages of XP lie in its ability to enable minor, frequent system releases that result in incremental development. Furthermore, Rodríguez (2018) suggests an increase in the development team productivity due to the rapid feedback the end users provide since they are part of the development team. Also, the system's quality using XP is greatly enhanced due to establishing test cases early in the development cycle and code refactoring. Tessem (2003) and Mannaro et al. (2004) also highlight that XP's advantages are enhanced by pair programming. Pair programming facilitates constant learning by team members, enabling project deadlines to be met and a better quality project delivered to the client (Tessem, 2003). Developers feel more productive when working in pairs (Mannaro et al., 2004; Petersen & Wohlin, 2009). However, William (2010) and Abrahamsson et al. (2017) also cite some disadvantages of XP. These are the skilling of team members on the technical aspects of test-driven development, and although it is adequate to have an end-user as of the development team, it is also time-consuming (Martin et al., 2004). Furthermore, creating an integrated environment for continuous testing is challenging, especially when it needs to run on different platforms (Karlstrom & Runeson, 2005).

2.5.5 Scrum

XP concentrated on more of the technical aspects of software development (pair programming, testing, refactoring, coding standards and simple design. XP lacked the project management aspects of software development. To address this, Hirotaka Takeuchi and Ikujiro Nonaka introduced the term Scrum in 1986 in the article, 'The New New Product Development Game" (Takeuchi & Nonaka, 1986). Jeff Sutherland further developed the Scrum concept, which Ken Schwaber later formalised (Sutherland & Schwaber, 2007). The Scrum methodology has its roots in the game of rugby, where a multidisciplinary team comes together from beginning to end to score a try in rugby. Similarly, a group of multidisciplinary software professionals (systems analysts, testers, programmers, etc.) together with the client comes together to envision requirements for a software project and break it down into increments that are developed iteratively by the team until the software project is completed (Sutherland & Schwaber, 2007). According to (Schwaber, 1997), speed and flexibility in the Scrum methodology are attained by having strategies incorporated into the process that enable the speedy development and deployment of functional software.

Takeuchi and Nonaka (1986, p. 1) state the following about speed and flexibility,

"This new emphasis on speed and flexibility calls for a different approach for managing new product development. The traditional sequential or "relay race" approach to product development exemplified by the National Aeronautics and Space Administration's phased program planning (PPP) system-may conflict with the goals of maximum speed and flexibility. Instead, a holistic or "rugby" approach-where a team tries to go the distance as a unit, passing the ball back and forth-may better serve today's competitive requirements."

According to Sutherland et al. (2012, p. 14), "Scrum is an iterative, incremental framework for projects and product or application development. It structures development in cycles of work called Sprints". Sprints are a key practice of Scrum, and Pressman (2010) defines sprints as a unit of work that must be accomplished within a specific timebox, and the timebox is a fixed, immovable time frame lasting 2 to 4 weeks. A flexible design and plan and a definition of what will be developed during the sprint are used to guide the development process (Sutherland et al., 2012).

Schwaber and Beedle (2002) suggest other practices that are implemented within the Scrum framework. These categories are the Scrum Roles, Product Backlog, Daily Scrum Meetings, Sprint Planning Meeting, and Sprint Review.

Scrum roles: The three roles incorporated into the Scrum framework are the Product Owner, Team, and Scrum Master. The product owner ensures that the developed software will have the desired benefit envisaged by the organisation. This is achieved when the product owner initially prioritises the requirements in the product backlog during the initial planning meeting and after that makes certain that any new requirements are added to the product backlog and prioritised. The Scrum team must consist of software professionals with diverse skills in analysis, development, testing, interface design, database design, architecture, etc. This cross-functional team must be self-organising and have the expertise to create software increments with every sprint. The scrum team is responsible for selecting a set of requirements from the product backlog to form the sprint. Critical to the success of the Scrum framework is the Scrum Master, who ensures that Scrum processes are followed and is also a liaison between the organisation, product owner, and Scrum team to ensure that there is meaningful collaboration and interaction (Sutherland & Schwaber, 2007). According to Cohn (2006), the Scrum Master ensures that the Scrum team focuses maximum on the Sprint. (Cho, 2008) summarises the responsibilities of the Scrum Master as ensuring that the development process is adequately managed and the development team has an enabling environment, resulting in optimum productivity during a Sprint.

Product backlog: The product backlog contains a list of prioritised requirements and the estimated time the Scrum team will take to develop. As customers need changes, brainstorm new ideas, or technical obstacles arise, the product backlog is continuously updated to reflect these changes. The time estimate of each item in the product backlog is essential in planning future sprints (Sutherland & Schwaber, 2007).

Sprint backlog: The sprint backlog lists the tasks developed during a sprint. The Scrum team updates the sprint backlog as requirements are completed during the sprint. A Sprint burndown chart can graphically represent the tasks remaining in a sprint. As tasks are completed, this chart is updated, enabling the Scrum team to assess the effort required to complete the Sprint. The name of this chart is derived from a downward-sloping graph that is created as the sprint progresses and will reach zero by the last date of the sprint (Sutherland & Schwaber, 2007).

Sprint Planning meeting: Each Sprint begins with a Sprint planning meeting. During the first part of the meeting, the product manager describes the highest priority requirements/features to the Scrum team. The Scrum team takes the features, breaks them into tasks, and allocates the time it will take to complete each task. A group of tasks are then allocated to a sprint (Sutherland & Schwaber, 2007).

Daily scrum: A short 15-minute meeting held at the beginning of the day in which the Scrum team discusses what has been achieved the day before, what obstacles faced by team members that prevented them from achieving their goal, and what will be completed today. These meetings also provide an opportunity for experienced developers to impact their knowledge of more inexperienced team members (Sutherland & Schwaber, 2007).

Sprint review: After a sprint, the product owner, scrum master, and team attend a sprint review meeting to determine what has worked and has not worked during the sprint. This will prevent them from making the same mistake in the next sprint and identify risks that must be overcome before the next sprint (Sutherland & Schwaber, 2007).

These characteristics are incorporated into the Scrum process, as illustrated in Figure 2.11. The process starts with requirements generated via input from end-users, customers, teams and other stakeholders. These requirements are maintained in the product backlog. Some requirements from the product backlog are selected to form the sprint backlog, which is then developed over a 1-4 weeks sprint cycle. During this sprint, there are daily Scrum meetings. Also, during the course of the sprint, there is product backlog refinement as a result of new changes that emerge as a sprint is in process. At the end of a sprint, if the product is deemed acceptable for implementation, it is delivered to the operations team for deployment into the operational environments. After that, a sprint meeting is carried out before the cycle begins again (Pressman, 2010).

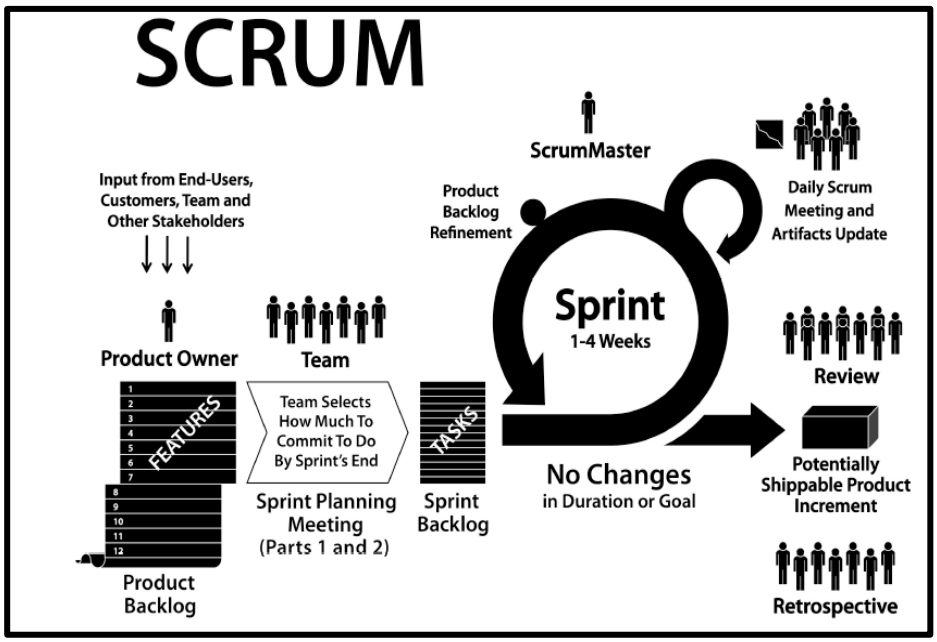

Figure 2.11: Scrum process model (Sutherland et al., 2012)

The advantages of Scrum are that software can be developed and tested quickly, and any changes can be easily amended in future iterations. Due to daily scrum meetings, there is clear visibility of project progress. Due to constant interaction between developers and the customer during project development, the overall quality of the project is better (Mahalakshmi & Sundararajan, 2013).

According to Sutherland (2005), the lack of documentation in Scrum is a disadvantage. Suppose this results in the incomplete documentation of the system's specifications. In that case, it will decrease the developers' productivity since they will be unsure of what needs to be developed. Sutherland (2005) also found that Scrum success is tightly aligned to teamwork, and if any team member does not perform, there is an excellent likelihood that the project will fail.

2.6 THE NEED FOR A DEVOPS INTERVENTION

The Agile software development process allows for continuous development of software. With the increasing demand for efficiency and effectiveness of software in the digital environment, quick deployment of software into the operational environment is necessary. This is not the case, and according to Jonker (2017), the optimisation of the operational phase during Agile development is often overlooked, resulting in Agile methodology not providing optimal time to market. Leite et al. (2019) further state that existing literature on Agile focuses very little on deployment practices. Humble and Farley (2010) state that deployment in agile

environments is predominantly manual processes, containing error-prone activities leading to time wastage due to debugging deployment problems. Therefore, the transition from development to production environment is frustrating and stressful in organisations (Leite et al., 2019). Besides manual deployment practices, IT divisions within large organisations typically distribute their workforce into different departments. It is common in these organisations to have a separate software development department and an IT operations department. This division of duties is essential because, apart from coding skills, various other skills are needed when developing and releasing software (Kristinsson, 2015). Although their approach to providing software to the organisation may differ, the operations and development department will always be a crucial part of the software development process (Kristinsson, 2015). Operations are responsible for implementing new and changed software into production and ensuring that the production system functions correctly at the appropriate service levels (Woods, 2016), while the development department frequently creates new features to meet the changing business requirements. Hence, development teams continually add new functionality to the software, whereas operation teams require less frequent software changes to achieve more stability and security in the operational environment (Aiello & Sachs, 2016; Punjabi & Bajaj, 2016).

Table 2.2 illustrates the different values and goals between developers and operations, resulting in the conflict mentioned above.

Table 2.2: Different values and goals between developers and operations (Subramanya, 2016)

Ops / ITIL Values	Agile Dev Values
• Procedure Driven • Stability • Availability/Uptime • Controlled/Frozen Environment • Infrequent Updates ***Results in:*** • Long Lead Time • Limiting the # of Changes	• Business Driven • Responsive to Change • Real-Time • Constantly up-to-date environment • CI / CD Environment ***Results in:*** • Short Sprints (2-3 wk) • Lots of small changes
• Infrequent Deployments	• Frequent Deployments

These different priorities result in conflict between the operations and development teams since the development team wants frequent changes to the production system. The operations team wants a stable production environment by preventing changes that will lead to downtime in the production environment. This result is known as "throwing the code over the wall", where developed features are thrown over the wall by the development team to the operations team, which results in the development team thinking their work is complete. However, when the code is uploaded into the production environment, various errors can result, leaving each division blaming the other for the errors. This creates a bottleneck since there was a lag between when the software was delivered to the operations team and when it is deployed into the operational environment, as illustrated in Figure 2.12.

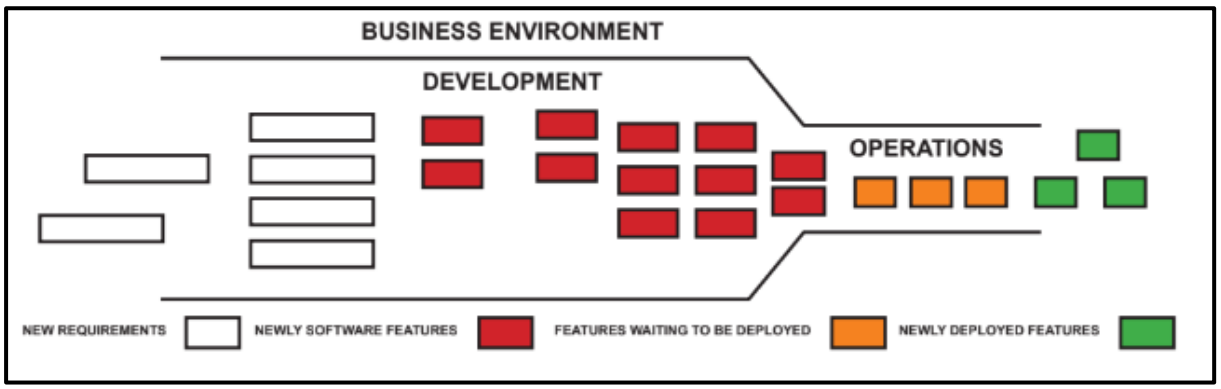

Figure 2.12: Bottleneck approach to software development and deployment (Own Work)

Hence, stakeholders do not immediately experience the benefits of new software releases. While working with operations and development teams on various projects, Patrick Debois realised a better way to handle conflicts between operations and development, resulting in the birth of DevOps (Liu & Zhou, 2017). DevOps, the combination of development and operations, is a natural extension of Agile development. The best practices and guiding principles applied to Agile developers are applied across the organisation in DevOps (Bradley, 2015). This prevents the so-called "throwing the code over the wall mentality", where programmers pass completed code to operations for implementation and do not take any responsibility for errors that might result during execution. It promotes the principle of "you code it, you own it", ensuring that a team of professionals (developers, security specialists, testers, operations) work together to ensure the code's success. They are breaking the wall of confusion between developers and processes, resulting in improved collaboration and faster resolution of issues. According to the Puppet_Dora (2017), organisations attained deployment of code 46X more frequently, lead time for changes 440X faster, mean time to recover 96X more quickly and

change failure rate 5X lower in organisations that implement DevOps. Research also indicates a better alignment of IT responsiveness and capabilities to business needs, smaller, more frequent software releases, and reduced effort with software development, transition and operation. There is also improved time to market, improved quality of code, improved quality of software deployments, reduced cost of product iteration and delays, instil a culture of communication and collaboration, improved productivity, and improved visibility into IT requirements and processes (Balachandran et al., 2016; Colavita, 2016; Virmani, 2015b).

There has also been a significant shift in the types of systems that customers require. It's the transition from Systems of Record (systems that function as a DB of massive amounts of data/transactions) to Systems of Engagement (systems that customers can access directly to interact with the business). This new requirement of Systems of Engagement requires an intense focus on user experience, speed of delivery and agility, which aligns with the DevOps strategy of continuous deployment. (Sharma & Coyne, 2015).

2.7 WHAT IS DEVOPS?

Olszewska and Waldén (2015, p. 2) describe DevOps "as a software development method that combines quality assurance mechanisms with IT operations within software engineering practices". Farroha and Farroha (2014, p. 288), suggest that "DevOps is a business strategy to overlap traditional operational and developmental lanes to create an environment that continually improves operations through cross-pollination of developers and operators".

Bass et al. (2015) further define DevOps as "a set of practices intended to reduce the time between committing a change to a system and the change being placed into normal production while ensuring high quality".

Jonker (2017, p. 32) provides the following comprehensive definition of DevOps as "… cultural movement that breaks silos between the business, development and operations department, combined with a number of service development practices regarding people, process and technology that enable rapid development and delivery of services".

The definitions highlighted above show that there is no single definition for DevOps. Still, the core component of all these definitions is the coming together of software developers and operations to build software that can add value to stakeholders, allowing companies to meet their strategic objectives. The main aim of DevOps is the collaboration of Dev and Ops that helps IT organisations have more flexible reactions to changes in the business environment

(Brunnert et al., 2015; Sharma & Coyne, 2015). DevOps permits a more frequent roll-out of new features and bug fixes in minutes, hours or days (Brunnert et al., 2015). Figure 2.13 illustrates how merging development and operations into one team allows for more features to be deployed into the business environment.

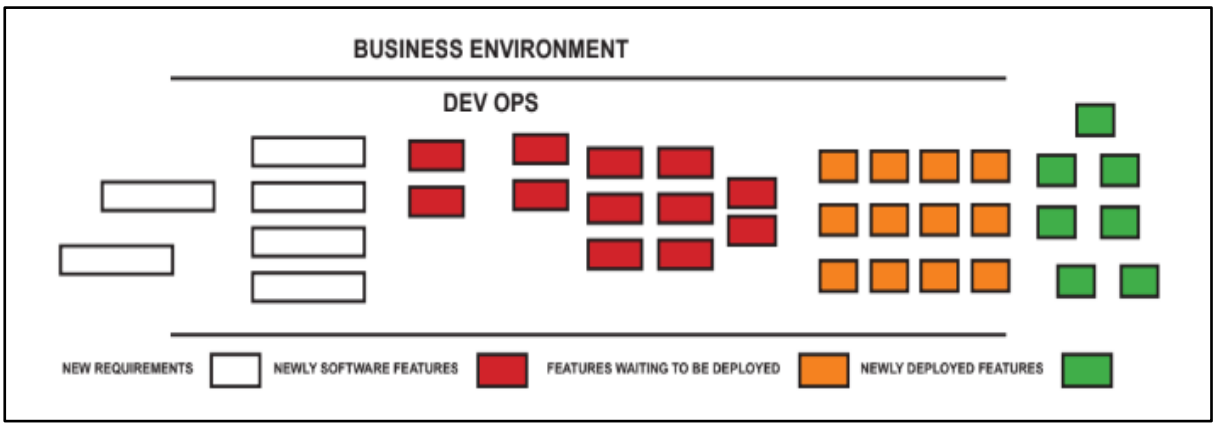

Figure 2.13: DevOps environment showing increased deployment of new features (Own Work)

Wahaballa et al. (2015) noted four problems that DevOps can alleviate:

Fear of change: Once an application is deployed, any change to this application to add new features or fix a bug takes a long time to be implemented. This is due to stakeholders being afraid of change, which results in bureaucratic management processes affecting changes.

Risky Deployments: Deployment is about making the application's services available to the end-user. Developers and Operators are worried because neither team is confident that the system will work in the real environment.

Blame Game: Errors detected by the end-user or system administrator lead to the developers and operators blaming each other.

Isolation: Traditional project team management involves Development (programmers, tester, and Quality Assurance) and Operations (DB administrators, system administrators, network administrators, and operators.) working in divisions which act like silos because they are independent of each other.

2.8 BENEFITS OF DEVOPS

Faustino et al. (2022); Nagpal and Shadab (2014); Peuraniemi (2014) classify the benefits of DevOps into technical and business performance benefits. In terms of technical benefits, DevOps enables the software development team to learn about problems much earlier due to

regular feedback, continuous testing and continuous delivery (Peuraniemi, 2014). Due to software being regularly integrated with extensive testing, fewer complex problems are discovered. DevOps teams are cross-functional, resulting in less time to fix errors since errors are solved as a cross-functional team and not passed from one division to another (Bou Ghantous & Gill, 2017; Nagpal & Shadab, 2014). Faustino et al. (2022); Katal et al. (2019) further state that due to automation, there is a decrease in the number of manual tasks performed and overall better system quality and reliability.

Business performance benefits are reduced time to market of software, helping customers gain competitive advantage, customer loyalty and better revenue (Faustino et al., 2022; Nagpal & Shadab, 2014). Shorter development cycle because of increased collaboration and communication between operations and development teams. A shorter development time increases the release velocity allowing for increased frequency for the deployment of code into the production environment. Organisations can also recover investments faster due to software's early release, allowing for a faster return on investment. Organisations perform better since they can deploy a business-critical application without disrupting services. Improved quality in the software project due to the smaller deliverables that are easier to debug and test resulting in a more stable overall system. Reduction in IT wastes using the DevOps approach since there is shorter lead-time, more planned work and properly coordinated deployments. Also, less waste reduction because of the high rate of success for deployments.

2.9 CHALLENGES TO DEVOPS ADOPTION

Some challenges organisations face in implementing DevOps are cultural shifts and a lack of skills and management support. Although DevOps is meant to change the culture in organisations by breaking down barriers and increasing collaboration, this culture shift becomes a challenge to implementation since many of the different teams within the software development environment are accustomed to their processes and ways of working, resulting in resistance to change. This resistance to change is a result of deep-seated culture within organisations and is one of the main challenges to DevOps adoption (Bucena & Kirikova, 2017; de França et al., 2016; Khan et al., 2022; Luz et al., 2019; Riungu-Kalliosaari et al., 2016; Rowse & Cohen, 2021).

Due to the recency of DevOps, it is problematic to find software development professionals with the experience, skills and knowledge to instil and support DevOps practices within

organisations (Khan et al., 2022). According to Senapathi et al. (2018, p. 8), the technical skill needed is not only coding but also "understanding infrastructure and its setup, deployment, post-deployment monitoring, infrastructure problem solving, and skills in using the supporting tools". Khan et al. (2022) further state that development personnel like to concentrate on their area of expertise and are unwilling to learn other skills.

Lack of management support for DevOps adoption poses a considerable obstacle to the successful adoption of DevOps. Management is the critical driver of change within an organisation, and since DevOps results in restructuring the way software development is carried out in an organisation, if management does not support this process, it will be destined for failure. Furthermore, DevOps needs various resources to be successful, and management needs to invest in these resources for DevOps adoption to be successful (Bucena & Kirikova, 2017; Khan et al., 2022). Achieving security in DevOps is also a challenge. This is due to traditional security methods that cannot keep pace with the speed needed in a DevOps environment (Desai & Nisha, 2021; Myrbakken & Colomo-Palacios, 2017). Verizon also reports 'that nearly 70 percent of the data breaches they have investigated are due to attackers targeting vulnerable web applications, and risks are amplified as organisations move to DevOps without implementing proper security practices" (Checkmarx, 2020, p. 2).

2.10. THE WORKINGS OF DEVOPS

Sharma (2014), points out how DevOps works, in four steps:

Develop and test against production-like systems: The aim is to allow development and quality assurance teams to develop and test compared to systems that act like the production system, so they can see how the system performs before it is ready for deployment. It enables the operations team to note well in advance how their environment will perform when it supports the system, therefore permitting them to create a fine-tuned, application-aware environment. This concept moves operations earlier in the development lifecycle.

Deploy with repeatable, reliable processes: This step allows development and operations to support an Agile/Iterative development process to production. Automation is crucial to creating iterative, regular, repeatable and dependable processes. Frequent deployments allow the teams to test the deployment process, minimising the risk of deployment failures.

Monitor and validate operational quality: This step shifts monitoring earlier in the production life cycle. It enables automated testing to be done early and frequently in the life cycle, allowing

for observing the system's functional and non-functional features. Quality metrics should be gathered and analysed whenever a system is deployed and tested. Regular monitoring offers early warning signals about operational and quality matters that may occur in production.

Amplify feedback loops: This concept enables organisations to respond rapidly and make changes. It entices organisations to generate communication networks that allow stakeholders to access and act on feedback.

2.11 KEY PRINCIPLES OF DEVOPS

The working of DevOps is infused into the framework by four key principles:

Continuous integration (CI)

Previously, software developers should work in isolation for a lengthy time in code generation and would only incorporate the code of different individuals after completing their requirements segment. This made the merging of code difficult and allowed for the proliferation of bugs for an extended period without being detected, thereby delaying software release to clients (Hüttermann, 2012; Mohammad, 2016). CI is a practice in software engineering whereby team members integrate and merge their code several times a day into a single shared repository (Fitzgerald & Stol, 2017; Karamitsos et al., 2020). This enables the early identification and correction of bugs resulting in an earlier release of the software (Bosch, 2014; Shahin et al., 2017). Once code is uploaded into the repository, automatic build tools are used to test the code, resulting in the early identification of errors and enabling developers to access the most recent corrected code (Vaasanthi et al., 2017). CI is critical in the DevOps process since it endeavours to add speed, agility and security to the software development process (Yarlagadda, 2018).

Continuous delivery and continuous deployment

Continuous Delivery (CDE) follows CI as the next step in ensuring that the project is prepared for release to development, staging or production environments when requested (Humble & Farley, 2010). By using automated testing and quality checking, CDE aims to ensure that the software product is always ready to be deployed into production (Weber et al., 2016). CDE provides the benefits of reducing deployment risks, lowering costs and providing a faster turnaround time for receiving feedback from users and clients (Chen, 2017). Furthermore, it enables a business to add more functionalities to the application in a shorter frame, meeting the demands of users and catering for the dynamic changes that occur in industries (Yarlagadda,

2018). After the software is in a production-ready state, manual processes occur to push this software into production. However, continuous automation deployment (CD) practices push production-ready software into the production environment. CD uses automation to reduce the complexity of the deployment and operation processes. This plays a vital role in bridging the gap between development and operations and is a crucial practice to the success of DevOps (Bass et al., 2015; Newman, 2015; Shahin, 2015).

Continuous testing

Although general testing in software development assists in determining the quality of the software product and whether the software is meeting its desired goals, it is unsuitable for continuous integration due to testing being conducted manually and without the automation of test cases. Often this manual process takes a lot of time and effort prolonging the implementation of the software to the operational environment (Virmani, 2015a; Xuan et al., 2021; Zaib & Lakshmisetty, 2021). Hence it does not provide a process that allows for the rapid checking of quality and the fast error detection rate that is needed during continuous integration. Continuous testing can help alleviate this problem by decreasing human intervention leading to faster testing and feedback (Fitzgerald & Stol, 2014; Zaib & Lakshmisetty, 2021). In DevOps, faults in software artefacts are detected using continuous testing techniques. In most software process models prior to DevOps, testing was a dedicated phase within the lifecycle, typically occurring towards the end of the lifecycle. However, in DevOps, testing starts from the beginning and continues through the lifecycle, with constant collaboration between the development and operations teams during testing. This collaboration leads to increased software quality, reducing the likelihood of software failure in the operational environment (Faber, 2020).

Continuous monitoring

To guarantee that applications are stable and have the functionality to meet client needs, applications must be monitored continuously (Österberg, 2020). Continuous monitoring is a set of practices that the DevOps team use to monitor applications for the early identification of problems (Hüttermann, 2012; Sharma & Coyne, 2015). By continuously monitoring applications throughout the project development lifecycle and while in operation, performance degradation and infrastructure problems that prevent the optimal performance of the application can be identified. Furthermore, business logic errors can be detected early and resolved (Hüttermann, 2012). According to Teixeira et al. (2020, p. 12), “effective monitoring

is essential to allow DevOps teams to deliver at speed, to get feedback from production, and to increase customers' satisfaction, acquisition and retention".

2.12 DEVOPS TOOLS

DevOps aims to deliver high-quality software within a short cycle time. Tools are used for this high degree of automation. These tools are used during the different phases of development, as stated by (Ebert et al., 2016). Figure 2.14 illustrates the different tools that support different phases of the DevOps process.

Build Tools

Automation in the build phase is critical for the DevOps environment, and build tools are essential to achieve this. These tools handle the software development cycle. It comprises compiling code, handling dependencies, producing documentation, running tests, or deploying a system to diverse environments. An example is Apache Ant which has become a standard tool for building applications (Akshaya et al., 2015; Wiedemann et al., 2019).

Continuous-Integration Tools

CI tools integrate developers' work as early and frequently as possible to enable the system to be continuously tested. Hardware challenges arise because building and testing software on target hardware components are not always possible and may require simulation. An example is Bamboo from Attlasian (an enterprise software company that develops products for software developers, project managers, and content management) (Akshaya et al., 2015; Wiedemann et al., 2019).

Configuration Management Tools

CM tools try to lessen production provisioning and configuration maintenance while recreating the production system on the development machines. Machine-to-machine communication is critical for continuously delivering embedded systems, and the IoT architecture is necessary. An example is Puppet, whose approach to infrastructure provisioning is about describing the system's intended end-state (Akshaya et al., 2015; Wiedemann et al., 2019).

Logging Tools

Logging is an alternative for debugging, and even though programmers are recommended to debug, logging is necessary. Logging is about keeping traces in an application. The regulation is to log everything and establish the log level (fatal, error, warn, info, trace, or debug). An example is Loggly which provides a cloud-based service to accumulate log-data from applications, platforms and servers in real-time using open standards like HTTP. The configuration is easy and permits the creation of custom performance (Akshaya et al., 2015; Wiedemann et al., 2019).

Monitoring Tools

These tools enable the organisation to recognise and resolve IT infrastructure issues before they influence core business processes. They monitor system features like CPU load, RAM, network traffic statistics, memory consumption and availability of free disk space. They constantly monitor the system's health and alert the administrator if any one of the features creates problems to take essential corrective measures. An example is Nagios, a popular open-source tool for monitoring IT infrastructures, like end-user stations, IT services, and active network components. They also provide trending and capacity-planning graphs that allow organisations to design infrastructure advancements (Akshaya et al., 2015; Wiedemann et al., 2019).

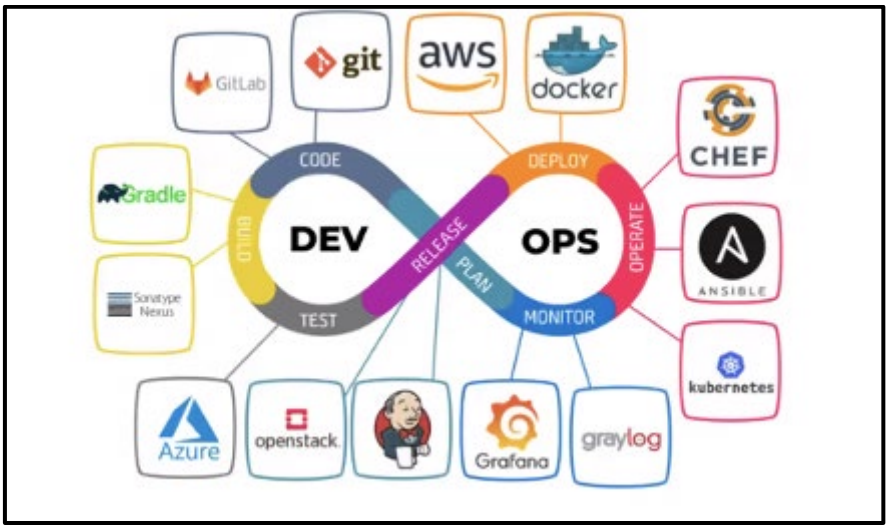

Figure 2.14: Popular DevOps tools (Altun, 2023)

2.13 FRAMEWORKS FOR DEVOPS IMPLEMENTATION

Hussaini (2014) presents the SNAC framework for implementing DevOps to strengthen the communication and organisation between development and operations teams. The SNAC

refers to Stakeholders, Needs, Alterable, and Constraints. A stakeholder in a software engineering project is an individual or group of people who are concerned with the performance of the software. Needs refer to the requirements that each stakeholder seeks to accomplish through the software. Alterable refers to what the program can change at a later time. Constraints are things that an organisation or program is unable to change. The SNAC framework aims to implement DevOps by analysing the software to be developed to generate a set of comprehensive objectives for the development and operations. The SNAC framework can be used to find different dimensions of a problem and common objectives between development and operation. The SNAC process includes the following steps:

1) Prepare a description of the software to be developed
2) Identify vital elements from the description
3) Classify the key elements in terms of the SNAC framework
4) Develop a set of objectives that satisfy the needs of the stakeholders
5) Validate and understand objectives
6) Identify success factors of the objectives

The SNAC framework enables the harmonisation between the development and operations teams, leading to faster and more efficient software releases and more motivated teams with improved communications.

Wettinger et al. (2014) suggest that the TOSCA model can be used for the implementation of a DevOps strategy. With the increase in the number of cloud applications available, the Topology and Orchestration Specification for Cloud Applications can be used to develop cloud-based applications (Wettinger et al., 2014). A key aspect of the DevOps strategy that is mentioned is to make the collaboration of development and operations teams more flexible. One way this can be achieved in a cloud-based environment is to automate most deployment and management tasks. The TOSCA model fits in with the DevOps strategy and aims to enable this flexibility, allowing for more efficient releases of cloud-based applications. The goal of this model is to provide development and operations teams the ability to better manage cloud applications, as well as allow for a more efficient portability of cloud applications.

One of the most used models for implementing DevOps is the CAMS model (Akbar et al., 2022; Perera et al., 2017; Rütz, 2019). CAMS (Culture, Automation, Measurement, and Sharing) was created by DevOps pioneers John Willis and Damon Edward. The culture element looks at the environment that allows for the interaction between development and operations teams and provides them with the ability to look forward to change. The automation aspect

looks at the smoothness of the workflow and the sharing of ideas between the two teams. The measurement element refers to the feedback provided that allows for improvements to be made. Sharing looks at how the discovery of new tools, experiences and success enablers or inhibitors can be shared with the organisation, increasing the success of DevOps. Perera et al. (2017) mention that the CAMS model is suitable for implementing DevOps as the four factors in the CAMS model can also be seen as pillars of the DevOps strategy. A study by Yarlagadda (2019) found that practising DevOps with the CAMS model increased software quality.

2.14 CURRENT APPLICATIONS OF DEVOPS AND ITS SUCCESS

Many large organisations such as Amazon, HP, Etsy, Netflix and Adobe have incorporated DevOps into their software development process. According to Null (2022), Amazon had a historical problem of not knowing how much hardware to purchase to cater to the influx of customers during peak buying periods. This resulted in Amazon over purchasing hardware which resulted in 40% of server capacity being wasted during non-peak periods. They adopted DevOps and Amazon Web Services (AWS) to overcome this problem. This resulted in capacity automatically being scaled up or down. Furthermore, there is a more continuous deployment of code, with code deployments occurring in 11.7 seconds on average.

Netflix moved from a company that posted movies on DVDs to a company that streams movies to customers. For Netflix streaming to be successful, the company had to adopt a DevOps culture in which employees collaborated to create new tools to manage the cloud infrastructure. The adoption of DevOps enabled them to innovate more rapidly, pushing new features more quickly, thereby improving customer satisfaction and increasing the subscriber base. Engineers at Netflix can carry out code deployments thousands of times daily (Dobra, 2021; Null, 2022). In order to provide needed features to customers more frequently, Facebook employed a DevOps philosophy. The company can provide more frequent updates and rapidly refresh its mobile apps by implementing new software tools and changing policies to give the developer code ownership. Code ownership enabled the developer to be responsible for every line of code from development to production (TechTarget, 2017). HP's problem was the lack of deployment and poor turnaround time for testing. Although it had around 400 developers worldwide, it only released software twice a year. Additionally, testing was time-consuming due to the manual methods employed. To alleviate these bottlenecks, HP adopted CI/CD and test automation. These DevOps practices resulted in 100 to 150 code commits per day and 100 000 code changes daily (Gene, 2014). Initially, Etsy had slow deployments (twice a week) due to the siloed teams not working well together. The company changed to providing

developers with code ownership, meaning they were responsible for the code from development to performance and uptime. This resulted in deployment occurring 60 times daily (Dhaduk, 2022).

2.15 EMBEDDING DEVOPS INTO SCALED AGILE FRAMEWORK (SAFe)

According to Reifer et al. (2003), implementing agile practices is more likely to succeed in small organisations with small teams than large organisations with many development teams. This is due to the difficulty in coordinating large teams and resistance to organisational change in large enterprises (Cohen et al., 2004; Dybâ & Dingsoyr, 2008). When a large project needs to be undertaken, the project is split into smaller components with each team working on a component. Due to the many teams working on a single project, coordination becomes an issue. Furthermore, when a culture is embedded in an organisation and suddenly changes, this can lead to resistance from employees within the organisation. This problem is intensified in larger organisations, leading to poor adoption of Agile since successful Agile adoption requires change in the organisational culture (Misra et al., 2010). Scaled Agile (Figure 2.15) is a model that implements agile practices in large organisations with hundreds of development teams (Leffingwell, 2007). Known as the Scaled Agile Framework (SAFe), this framework is selected by an organisation seeking methods for scaling agile development across the organisation. The adoption of agile practices is common in many organisations, and organisations are moving away from whether to adopt agile practices to how these methods can effectively adapt and scale these practices.

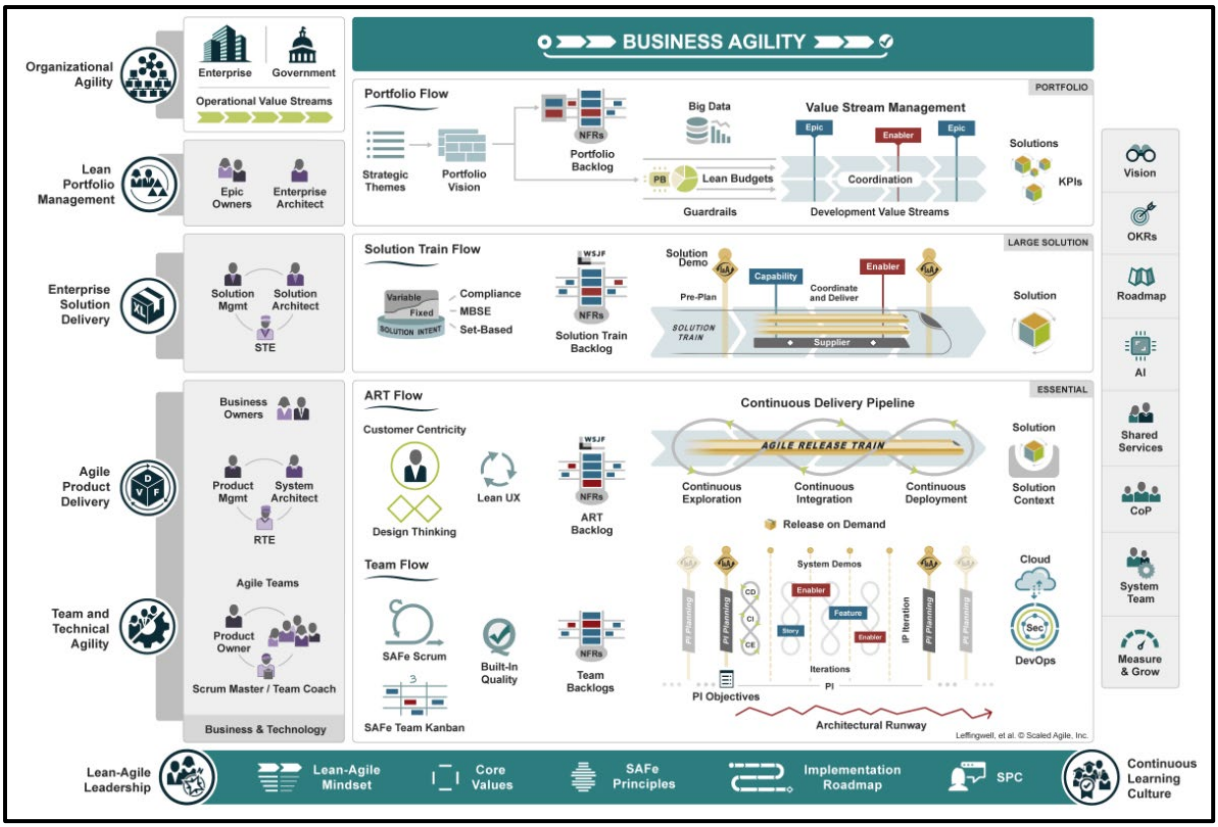

Figure 2.15: Full SAFe 6.0 (Scaled_Agile, 2023)

Various benefits, such as faster time to market, better productivity and quality, reduced risks, and project costs, can all be attributed to SAFe (Stojanov et al., 2015). SAFe has seven core competencies embedded into its framework to enhance business agility. SAFe is underpinned by four different frameworks configured to manage different organisational environments.

An overview of the SAFe frameworks (Scaled_Agile, 2023) is provided for reference.

Essential SAFe: This consists of the SAFe Team and the SAFe Program levels. SAFe guides the coding part of system development at the team level. At the program level, SAFe guides the operations activities that enable business value. An important component of the Essential SAFe is the agile release train (ART), which guides cross-functional teams in delivering the development and operations value streams.

Portfolio SAFe: An enterprise-wide plan that uses value streams (a term used to describe an enterprise-wide strategy to develop products, services or software systems) that provide value to the customer. The Portfolio SAFe is aligned with the organisational imperative to identify strategies that enable product differentiation in the marketplace and to ensure competitive

advantage. Leffingwell (2010, p. 43) refers to these strategies as “a set of investment themes”. These investment themes are achieved in the form of “epics”, a term used as a high-level descriptor of customer need and translates to a software development initiative. These epics are maintained in a portfolio backlog. One of the key role players is the Enterprise Architect, a person (or group of persons) who manages the portfolio backlog and works across programs (from the Essential SAFe) to provide technical direction to ensure the portfolio's outcomes are optimally achieved. The portfolio SAFe is a scaled-up business version of Agile Software Development Methodology used by large organisations with multiple Agile teams.

Large Solution SAFe: used for developing complex enterprise-wide solutions; typically used for government and military systems and requires multiple ARTs

Full SAFe: a SAFe configuration that is the most comprehensive version of the framework and provides support for organisations that build and maintain large, integrated solutions and require extensive collaboration across the organisation to include stakeholders that function at the SAFe Team, Program, Large Solution and Portfolio levels of the framework

DevOps is an integral part of SAFe. “SAFe® enterprises implement DevOps to break down silos and empower each Agile Release Train (ART) to deliver new features to their end users continuously.” This enables the separation between development and operations to be drastically reduced, allowing for an automated continuous delivery pipeline of software. Without excessive operations support, this method effortlessly defines, implements, and delivers solution elements to the end user. The benefit is a production environment that is more flexible and stable, enabling the delivery of value more frequently (Scaled_Agile, 2022, p. 1).

2.16 THE INTERCONNECTION OF INDUSTRY 4.0 AND DEVOPS

Recently, “Industry 4.0” has been a hotspot for industries. Industry 4.0, which is the fourth industrial revolution, will significantly influence various industries by enabling a concept known as intelligent manufacturing. A digital and intelligent factory is achieved using ICT, industrial technology and Cyber-Physical Systems (CPS), resulting in a more digitalised, information-driven, customised and green manufacturing environment. The goal of Industry 4.0 is to build a highly flexible production model of personalised and digital products and services, with real-time interactions between people, products and devices during the production process. During the first three industrial revolutions, humans observed and developed mechanical, electrical and IT, improving industrial productivity. The first industrial

revolution improved due to hydropower, steam power and machine tools; the second industrial revolution invented electricity and assembly lines. The third industrial revolution used electronics and IT to increase automation further. The fourth industrial revolution is emerging, led by the Internet of Things technology to integrate the real world with the information age for future industrial development (Zhou et al., 2015).

Hence, Industry 4.0 entails interconnected machines, plants, and IT systems in manufacturing companies. This substantially enhances the significance of software in Industry 4.0 and was found to be a significant hindrance in the adoption of Industry 4.0 (Hasselbring et al., 2019) Therefore, Hasselbring et al. (2019) suggest the implementation of Industrial DevOps as a strategy to incorporate the methodologies and cultural aspects of DevOps into industrial production environments. The core principle of this technique involves the ongoing and systematic process of operating, observing, and improving the overall production environment. By using this approach, it is possible to seamlessly integrate all parties involved, various systems, and data through gradual and adaptable measures, allowing for swift modifications when necessary.

During Industry 4.0, many interconnected devices will form systems. This system will be multifaceted and implemented across multiple enterprises and regions. Therefore, problems with these systems will involve the collaboration of various departments (eg. technical, operational and maintenance) in an organisation and even different organisations. According to Cao and Zhang (2016, p. 1013), for large-scale systems to have efficient operation and maintenance, "it is necessary to implement the agile and lean thinking of DevOps, and use the cloud computing and automation tools and other technologies to achieve incident automated processing, to establish the unified support team of cross-technology, business and operations of different departments, to minimise delay in the transfer process flow, thereby improving the efficiency of the incident management and meet the operation requirements of Industrial 4.0."

Furthermore, during the 4th Industrial Revolution, embedded systems connected via the Internet required frequent update delivery, efficient release management, constant integration, and data exchange. To achieve this, traditional development methods are not appropriate. Wijaya et al. (2019) investigated the use of DevOps in developing embedded systems since it allows systems to be integrated more efficiently with a lower failure rate and faster corrections of errors. Results showed that the adoption of DevOps on embedded system development was successful, allowing for easier deployment and better monitoring during the operation phase.

For the manufacturing sector to survive during the 4th industrial revolution, it must achieve rapid and flexible change. In order to attain this, there needs to be a close union between the manufacturing and IT industries. This will entail IT rapidly developing and deploying software that is needed by the manufacturing industries (Park & Huh, 2018). Park and Huh (2018) compared the waterfall method to the DevOps approach in manufacturing System Integration projects and found that the DevOps approach resulted in much better quality and fewer human resources. They conclude that adopting the DevOps method in future manufacturing IT projects will result in faster and more efficient development and operation. This approach involves continuous and repetitive improvement and operation, achieved through the convergence of manufacturing and IT.

Fitsilis et al. (2018) also imply that Industry 4.0 will result in enterprises' radical digital transformation, necessitating new software development and deployment approaches. Agile development, Lean Product Management, Site Reliability Engineering and DevOps will form the core of these new development approaches.

2.17 SOFTWARE FACTORY AND DEVOPS

Watts (2020) states that "companies like Google and Netflix have set the gold standard in software development, with many updates and releases pushed daily to fix bugs, strengthen code, introduce new features, and handle unpredictable scaling." This approach to software development is called a software factory that quickly, easily and frequently churns out new software. Using the lean code approach, these software factories deploy high-quality software and features that rapidly benefit businesses. For a software factory to be successful, it needs to decrease the interaction time among developers so developers can concentrate on technical challenges such as the construction, deployment, monitoring and maintenance of automated pipelines and data security within the enterprise. Creating a DevOps environment where software developers and IT operators mutually cooperate is crucial to achieving this. By having this cooperation, functional silos are eliminated, and all team members are involved in the entire software lifecycle, from the design to deployment, so everyone in the team is responsible for the deployed code (Watts, 2020). Companies that are leading software factory implementation are transitioning from traditional waterfall-based methodology to one that is more dynamic, thereby removing silos within the organisation. Each team in the organisation becomes fully responsible for every facet of the software lifecycle, from designing to

developing and deploying, thereby reducing handover between teams. To achieve this, implementing DevOps into the teams is crucial (Shimel, 2018). Ravichandran et al. (2016) mention that the value proposition of IT is not limited to maintaining technological operations and occasionally providing enhancements over extended periods. It is focused on consistently generating business value through a state-of-the-art, efficient software production facility. DevOps, which emphasises the close communication between development and other IT organisations, while automating crucial application delivery procedures, has become an essential business necessity.

2.18 SUMMARY

This chapter presents the various software processes that can be used during software development, highlighting their advantages and disadvantages. Agile is a predominant process model for business systems that requires constant development of new features and upgrades to existing ones. However, this process model did not interact closely with the operations department, leaving a lag between the development of features and the implementation in the operational environment. To overcome this, DevOps was introduced to merge the development and operation teams to allow for greater collaboration, resulting in the more frequent release of features into the operational environment. Also, this chapter examines the impact that DevOps will have on software created in the future by discussing the role DevOps plays in the fourth industrial revolution and in the modernised software factory. As the world becomes fully digitised, software will be constantly needed. Therefore, DevOps will play a vital role as software development industries move from software development to software manufacturing by enabling collaboration across the entire software development lifecycle and automating the processes within this lifecycle. With any new innovation and technology, studying the factors that influence its acceptance and success is essential. Therefore, the next chapter will highlight the literature on acceptance and success factors in software development processes and derive an initial conceptual model to study these factors within the DevOps domain.

CHAPTER 3: THEORETICAL UNDERPINNING AND THE STUDY'S PRELIMINARY CONCEPTUAL MODEL

3.1 INTRODUCTION

As mentioned in Chapter 1, this study's main objective is to model the acceptance and success factors for DevOps. To achieve this, it is necessary to provide a theoretical lens through which a researcher can study these factors. Therefore, it is imperative to study the theoretical underpinning that supports this study. This theoretical underpinning can be documented in theoretical frameworks, and central to the theoretical framework is the concept of a theory. Fox and Bayat (2007, p. 29) define theory as "a set of interrelated propositions, concepts and definitions that present a systematic point of view of specifying relationships between variables with a view to predicting and explaining phenomena". Hence according to Kivunja (2018), a theory allows researchers to characterise what is being observed, comprehend and describe associations and to allow the researcher to understand human perceptions. A theoretical framework comprises a set of theories within a specific domain that serves as a foundation for the researcher to gather, analyse, and interpret results. Swanson and Chermack (2013, p. 122) explicitly assert that "The theoretical framework is the structure that can hold or support a theory of a research study". Sometimes, a theoretical framework may not encompass all the factors a researcher wants to investigate. In these instances, the researcher will derive a conceptual model combining constructs from existing frameworks and literature on the domain under study (Imenda, 2014).

The chapter first presents the theoretical background of acceptance theories. Furthermore, since this study also investigates the success factors of DevOps projects, this chapter will discuss the various literature on success factors in software development. The chapter will culminate in a preliminary conceptual model derived from the acceptance theories and the literature on success factors.

3.2. THE CONCEPT OF ACCEPTANCE

IS are known to be underutilised in many organisations. This leads to substantial financial losses and prevents the organisation from achieving the operational efficiency and competitive advantage envisioned by these systems. This underutilisation of IS has existed for many decades (Dillon & Morris, 1996; Lancelot Miltgen et al., 2013; Wang et al., 2006). The concept of user acceptance can explain the poor utilisation of IS. "User acceptance can be defined as the demonstrable willingness within a user group to employ IT for the tasks it is designed to

support" (Dillon & Morris, 1996, p. 6). In providing a suitable definition of technology acceptance, Louho and Kallioja (2006) defined technology acceptance as the way people perceive, adopt and use technology and mention that user acceptance is a critical factor in system success. Therefore, IS research in accessing the success or failures of IS forms the central tenet in acceptance studies. Davis (1993) also states that although systems will be implemented successfully, a lack of user acceptance is a common reason these systems fail. Taherdoost (2019) mention that great applications can be designed and developed, but if users are unwilling to use the application, the application will fail. Therefore, studying user acceptance is an integral part of the development and implementation of systems and determining the factors that influence acceptance will assist in increasing the level of usage and adoption. Since human interaction with technology is affected by various social and psychological factors, making it challenging to forecast human behaviour, various theories and models have been established to help predict acceptance and the factors influencing acceptance. The predominant theories are the Theory of Reasoned Action (TRA), Theory of Planned Behaviour (TPB), Diffusion of Innovation (DOI), Social Cognitive Theory (SCT), Technology Acceptance Model (TAM), Motivation Model (MM), Combined TAM and TPB (C-TAM-TPB), and Unified Theory of Acceptance and Use of Technology (UTAUT) and UTAUT2. Each of the models will be discussed in subsequent sections.

3.3 ACCEPTANCE THEORIES AND MODELS

3.3.1. Theory of Reasoned Action

Using previous research in social psychology, persuasion models and attitude theories, Martin Fishbein and Icek Aizen 1967 developed the theory of reasoned action (Ajzen & Fishbein, 1969). According to this model (Figure 3.1), a person's behavioural intention (BI) is determined by the constructs, such as attitude towards an act or behaviour and subjective norms. Attitude towards an act or behaviour is defined as "an individual's positive or negative feeling about performing the target behaviour" (Fishbein & Ajzen, 1975, p. 302) and subjective norm as "the person's perception that most people who are important to him think he should or should not perform the behaviour in question" (Fishbein & Ajzen, 1975, p. 302). Furthermore, a measure of BI will result from a voluntary act. This voluntary act is termed volitional control, meaning that a behavioural action can be executed whenever a user is willing to execute it. Therefore, factors such as skills, resources, knowledge and overcoming environmental obstacles outside an individual's voluntary control create conditions that the TRA model cannot

meet. Hence, in these cases, although the intention to carry out a task is strong, the individual may not be able to do the task due to factors outside their control (Fishbein & Ajzen, 1975). Therefore, researchers suggest that TRA is more behavioural than goal-oriented (Sheppard et al., 1988). Behavioural-oriented is when a behaviour is executed without a specific outcome of what that behaviour will achieve.

TRA model was used across numerous disciplines in various studies, such as dieting, genetically engineered foods and limiting sun exposure (Otieno et al., 2015). Several studies within the IS domain used TRA to establish an individual's intention to use technologies. Mishra et al. (2014) used it in determining green technology acceptance, Yousafzai et al. (2010) and Nor et al. (2008) in explaining internet banking behaviour, Liker and Sindi (1997) in studying expert system acceptance, Mykytyn and Harrison (1993) looking into senior management acceptance of strategic IS and Özer and Yilmaz (2011) researching the usage of IT by accountants.

A major limitation of TRA is that behaviour is under volitional control, meaning that an action does not occur spontaneously but is thought of beforehand. Due to this limitation, irrational decisions or behaviour cannot be described by this theory (Sheppard et al., 1988). In a voluntary environment, TRA only explains 30% of the variance (Venkatesh et al., 2003).

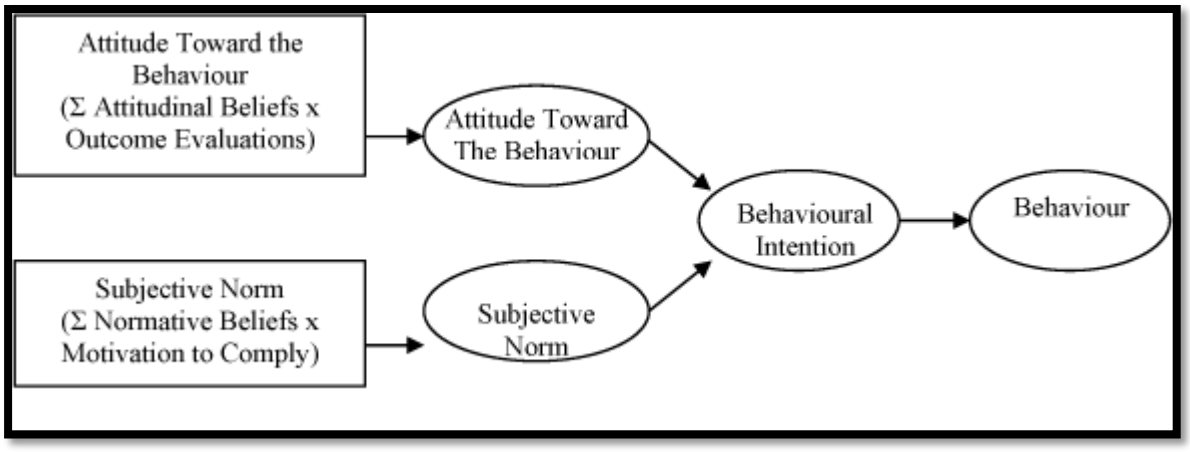

Figure 3.1: Theory of reasoned action (Fishbein & Ajzen, 1975)

3.3.2. Theory of Planned Behaviour

To overcome the weakness of incomplete volitional control in TRA, TRA was extended to create the theory of planned behaviour (TPB) (Figure 3.2). This was done by creating a new construct 'perceived behavioural control (PCB)' that evaluates a person's perception regarding the ease by which a behaviour can be performed (Ajzen, 1985). According to Ajzen (1991), PCB explains situations where a person does not have complete control over a behaviour. Control beliefs are a person's belief regarding the availability of factors such as financial status, time and skills (Ajzen, 1985; Sheppard et al., 1988). According to Bandura (1977) and Adams et al. (1992), factors such as these can affect a person's confidence, such as carrying out specific behaviours that will affect their actual behaviour. Behavioural intentions and PCB can predict actual behaviour. However, based on different circumstances, one will be a more dominant predictor than the other. For instance, if the behaviour is under the volitional control of an individual, then the BI will be a dominant factor in predicting actual behaviour. If the behaviour is not under the volitional control of an individual, then PCB will be the dominant predictor of actual behaviour (Armitage & Conner, 2001).

The theory of planned behaviour was used in various IS and technology studies. Examples of these are studies that investigated e-commerce (Pavlou, 2002), green IT (Akman & Mishra, 2014), e-government adoption (Ozkan & Kanat, 2011), mobile learning readiness (Cheon et al., 2012) and mobile banking adoption (Aboelmaged & Gebba, 2013)

Although TPB improves the prediction capability of behaviour over TRA, according to Armitage and Conner (2001, p. 1) it account for "27% and 39% of the variance in behaviour and intention, respectively". This is low compared to other acceptance models and therefore is a significant limitation when studying acceptance of technologies. Furthermore, having just four constructs to explain behaviour is insufficient to overcome this limitation. Taylor and Todd (1995b) extended the TPB by adding new constructs, resulting in the decomposed theory of planned behaviour (Figure 3.3). This model decomposes attitudinal, normative and control beliefs into multi-dimensional belief constructs. Attitude is decomposed into ease of use, perceived usefulness (PU) and compatibility. Subjective norms can be determined by peer influence and superior's influence. Resource-facilitating conditions and technology facilitating conditions influence perceived behavioural control. With this decomposition, DTPB moved

away from the generalised constructs of TPB into more specific constructs resulting in a greater explanatory and predictive power than TPB (Zahid et al., 2022).

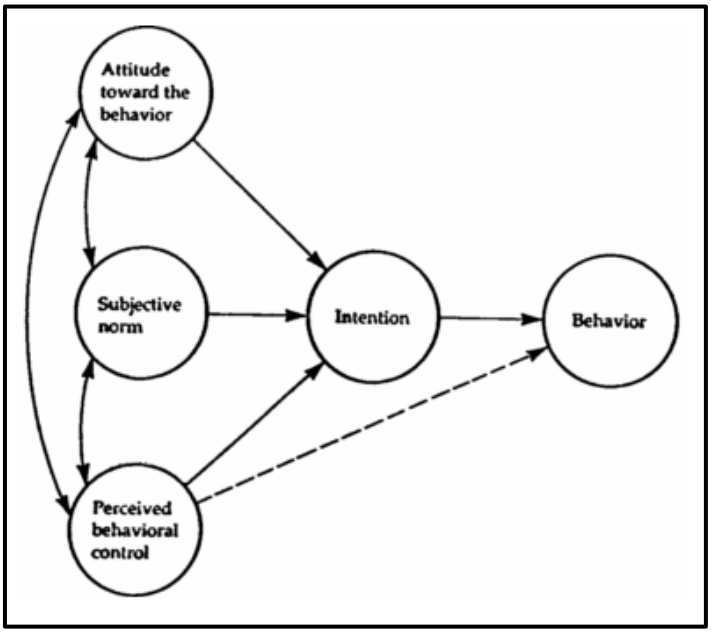

Figure 3.2: Theory of Planned Behaviour (Ajzen, 1991)

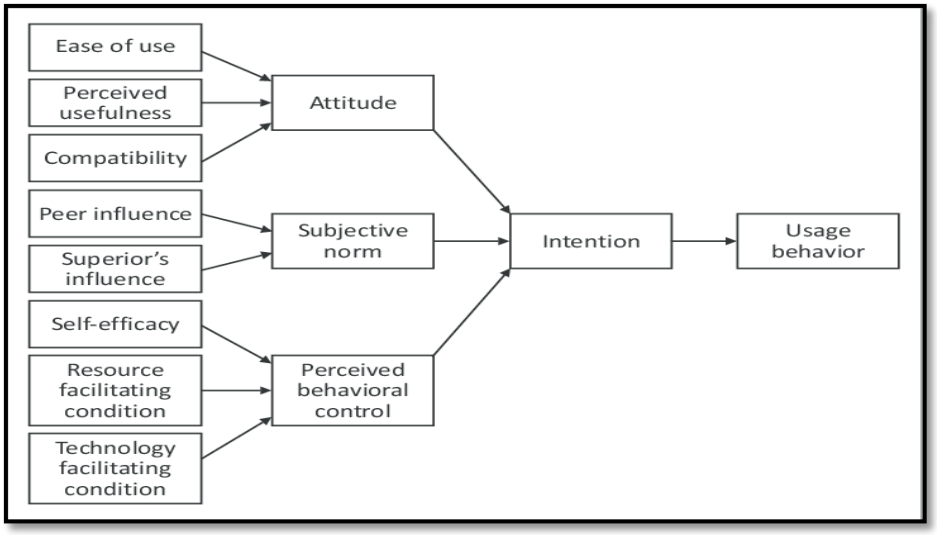

Figure 3.3: Decomposed Theory of Planned Behaviour (Taylor & Todd, 1995b)

3.3.3. Diffusion of Innovation Theory

In 1962, Rogers publicised the Diffusion of Innovation (DOI) theory (Rogers, 1962). This theory explains the diffusion of innovations in society and thus assists in determining whether organisations and individuals accept these innovations. (Rogers, 1983, p. 243) defines diffusion as "the adoption of an innovation over time by the given social system". This

diffusion process results in the acceptance or penetration of new ideas and innovations. According to Fichman (1992), numerous factors enable or inhibit technology adoption and implementation. These are classified as the innovation-decision process, attributions of the innovations and the innovators' characteristics.

Innovation Decision Process

The innovation-decision process is the process through which an individual or another decision-making unit first acquires knowledge about an innovation, enabling them to formulate an attitude towards it. The attitude about the innovation will lead to a decision to adopt or reject the innovation. If the decision is to adopt, then the innovation is implemented. Finally, there is the validation of the decision to implement the innovation. This process consists of five stages: Knowledge, Persuasion, Decision, Implementation and Confirmation (Rogers, 1983).

In the knowledge phase, the user acquires information regarding the new innovation by asking what, why and how questions. These questions are essential in understanding the innovation, how it works, and why it should be used. During the persuasion phase, the knowledge gathered by the intended user regarding the innovation is used to establish either a positive or negative perception of the innovation. Based on this perception, the innovation is accepted or rejected during the decision phase. The decision to reject can either be active or passive rejection. In active rejection, the potential user interacts with the innovation before rejecting it, whereas in passive rejection, the innovation is rejected without the potential user interacting with the innovation. If an innovation is accepted, it is put into practice during the implementation phase. Technical assistance during this phase is critical for the innovation to be successful. During the confirmation phase, the adopter reassesses the decision to adopt based on their experience with innovation and information from others. The decision to adopt can be reversed if the experience is negative and/or the information received about the innovation is negative (Rogers, 1983).

Attributions of the innovation

Relative advantage, compatibility, complexity, trialability and observability are crucial attributes that impact an innovation's adoption behaviour (Figure 3.4). Relative advantage is the PU an adopter will gain by adopting the innovation compared to the previous one. It refers to the perceived benefits of the latest innovation relative to the previous one, resulting in efficiency and economic gains (Rogers, 1983). Moore and Benbasat (1991) found that an innovation's perceived relative advantage positively influences the adoption rate. Compatibility is how the innovation matches their present and previous experience.

Furthermore, the innovation needs to align with the morals and belief system of the adopter. Complexity is the perceived ease of use (PEOU) of the innovation. Trialability is the use of the innovation for a short time, allowing the adopter to test the innovation before it can be adopted (Rogers, 1983). According to Agarwal and Prasad (1998), testing the innovation prior to adoption will increase the likelihood that the innovation will be adopted. Observability is whether the innovation provides benefits visible in the location it is implemented (Rogers, 1983).

Adoptors categories

There are different categories of adopters based on when they adopt an innovation. Rogers (1983, p. 22) defined the adopter categories as "the classifications of members of a social system on the basis of innovativeness". Individuals who adopt innovation as soon it is available are called innovators. These innovators have the financial assets and technical know-how to enable them to invest in the innovation and overcome any technical difficulty. Innovators consist of 2.5% of the population. Early adopters are the next group of individuals to adopt an innovation. These usually are opinion leaders who hold leadership positions and embrace new opportunities. Therefore, they do not need much convincing to adopt an innovation. Due to their status in the social system, they can drive others in society to adopt an innovation. Early adopters account for 13.5% of the population. The early majority are individuals who adopt an innovation after being provided with evidence of the success and effectiveness of the innovation. These are individuals who adopt an innovation before ordinary people in society. The early majority accounts for 34% of the population. The late majority are unconvinced about the innovation until it is tried and tested by most individuals.

Furthermore, there may be a lack of resources within this group, prompting them only to adopt an innovation proven to work and is effective. The late majority accounts for 34% of the adopters. Laggards are the most conservative individuals and, therefore, sceptical of change. Their resistance to adoption may also be due to a lack of resources. Laggards account for 16% of the adopters (Rogers, 1983).

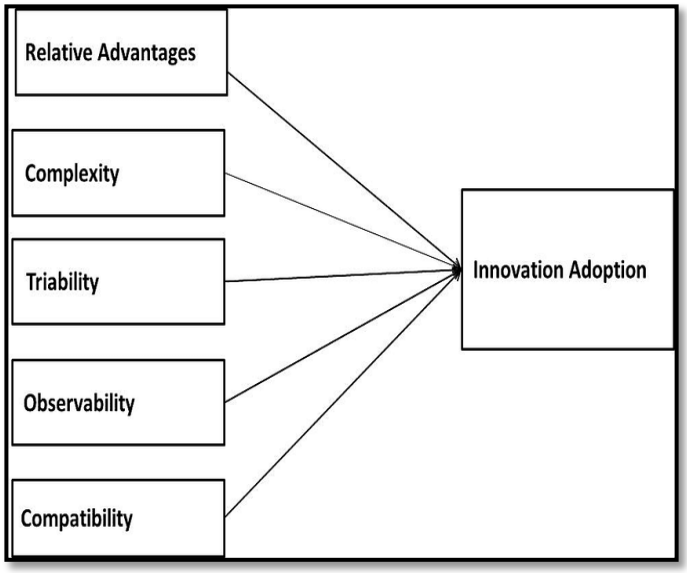

Figure 3.4: Diffusion of innovation theory (Rogers, 1983)

Within the domain of IS and technology, Al-Jabri and Sohail (2012) applied DOI to mobile banking adoption, Cheng et al. (2004) online games, Zhang et al. (2015) e-health innovation, supply chain technology, Faisal and Idris (2020), digital economy (Matyunina, 2019).

There are limitations levelled towards the Diffusion of Innovation Theory. According to Waterman (2004), the theory assumes that if someone understands an innovation, they will adopt it. The theory does not consider instances when the innovation is fully understood and it is still rejected. Clarke (1999) states that it is not as useful as other models in forecasting outcomes and, therefore, has less explanatory power of 38% (Venkatesh et al., 2003).

3.3.4. Social Cognitive Theory

Bandura introduced social cognitive theory (Figure 3.5) to provide a framework for understanding, predicting, and changing human behaviour. This theory posits that human behaviour can be explained by the interactions of personal factors, behaviour and the environment (Bandura, 1986b). The interaction of these factors is regarded as mutual, meaning that each factor is not only the cause but also the effect. For example, personal factors can influence an individual's behaviour and behaviour can influence individual personal factors. This bidirectionality is a key feature of SCT that distinguishes it from other

unidirectional models (Carillo, 2010). Human behaviour can also be determined by the behavioural determinants of an individual past, present and future behaviour. The environmental factors external to an individual play a role in encouraging or discouraging specific behaviour. The factors can be physical, weather, social, family, or co-workers. Personal factors consist of basic cognitive and affective human capacities such as self-efficacy, personal characteristics, expectations, self-regulation and reinforcement (Bandura, 1999; Ozmete & Hira, 2011).

According to Bandura, self-efficacy is a critical aspect of social cognitive theory. Self-efficacy is the perception of individuals to perform a task based on their skills. Directly related to the concept of self-efficacy is the factor of outcome expectations, where individuals will only perform a behaviour if they perceive that the outcome of performing the behaviour will benefit them (Bandura, 2001).

Hence behavioural change, according to the SCT is made possible by a personal sense of control. If people believe that they can take action to solve a problem, they become more inclined to do so and are committed to the decision.

In IS research, SCT was used to understand the behaviour of individuals being trained to use computers. Compeau and Higgins (1991) and Compeau and Higgins (1991) are credited with being among the first researchers to incorporate SCT into information system research when accessing computer technology and training success. They demonstrated that the inspiration of others positively influenced self-efficacy and outcome expectations. It was also found that self-efficacy plays a significant role in a person's decision to use a computer (Bolt et al., 2001; Compeau et al., 1999; Hasan & Ali, 2006). Compeau and Higgins (1991) mention that the strength of SCT compared to other models was its ability to use both self-efficacy and outcome expectation to understand technology usage. However, the explanatory power of SCT is only 37% (Venkatesh et al., 2003).

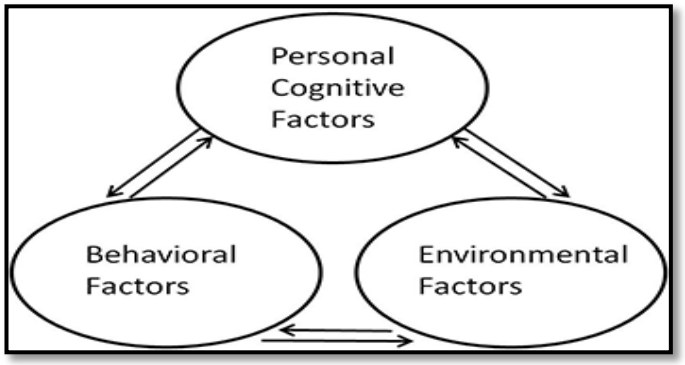

Figure 3.5: Social cognitive theory (Bandura, 1986b)

3.3.5. Model of PC Utilisation

In 1980, Triandis proposed the Theory of Interpersonal Behavior (TIB), which modified the same concepts and constructs as the theory of reasoned action. This modification is due to Triandis differentiating between beliefs that link emotions when an action is executed to beliefs that link the act to expected consequences in the future. He posits that people's feelings towards a behaviour, such as what they think they should do and its expected consequences, will determine BI. Hence behaviour will be determined by their habits, behavioural intentions and facilitating conditions.

Since Triandis model was not tested within the IS context, Thompson et al. (1991) tested a subset of the constructs to determine the factors that influence the utilisation of a PC by knowledge workers resulting in the Model of PC utilisation. Figure 3.6 illustrates the six factors that form the Model of PC Utilisation.

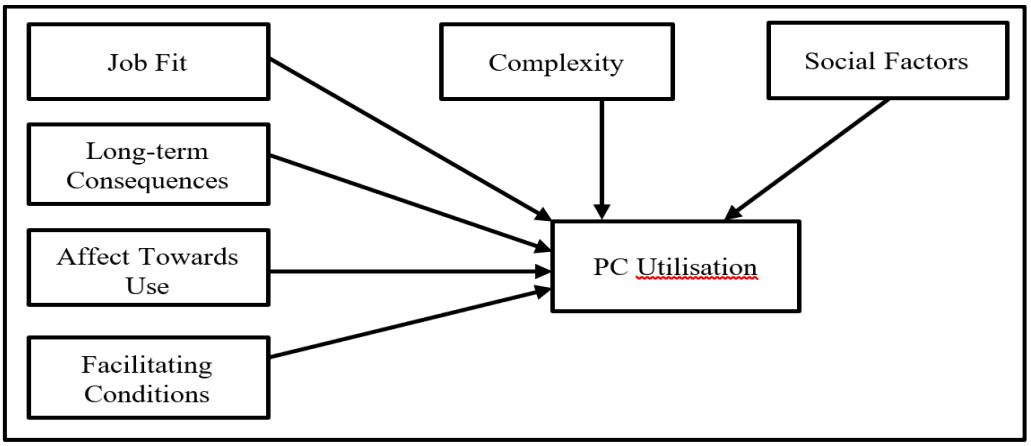

Figure 3.6: Model of PC Utilisation (Thompson et al., 1991)

a) **Job fit:** An individual believes that the technology can improve their job.

b) **Complexity:** measure the perceived difficulty of users in understanding and utilising an innovation

c) ***Social factors:*** are the interactions that users have with others that will shape their perception of technology. (Thompson et al., 1991, p. 126).

d) **Long-term consequences** are the benefits that are achieved in the future as a result of using technology. (Thompson et al., 1991, p. 129).

e) **Affect:** are the positive (joy, elation, etc) or negative (depression, disgust, etc) emotions experienced by individuals as a result of using technology (Thompson et al., 1991, p. 127).

f) **Facilitating Conditions:** provide a suitable environment for the users to utilise technology. Examples are providing technology and providing users with technical support to utilise the technology.

Apart from its low explanatory power of 37% in voluntary settings (Venkatesh et al. (2003), another limitation is that usage is based on the perception of the respondents, making it less accurate than it would be using actual usage statistics (Thompson et al., 1991).

3.3.6. Motivation Model

Using the concepts of extrinsic and intrinsic motivation that are part of the Self Determination Theory, Davis et al. (1992) studied the use of computers in the workplace. According to Ryan and Deci (2000), extrinsic motivation, which is external to a person, results in the performance of an activity to achieve a specific outcome. In contrast, intrinsic motivation exists internally in a person, which results in the performance of an activi ty for the inherent fulfilment of the activity. According to Davis, PU, PEOU and subjective norms are examples of extrinsic motivation. Enjoyment and computer playfulness are examples of intrinsic motivation. PU is defined as "the degree to which a person believes that using a particular system would enhance his or her job performance" (Davis, 1989), and enjoyment refers to the level of pleasure an individual feels when using a computer and this enjoyment is independent of the increased performance that might be expected (Carroll & Thomas, 1988; Deci, 1971; Malone, 1981). Figure 3.7 demonstrates the impact of extrinsic and intrinsic motivation on BI and the effect of extrinsic motivation on intrinsic motivation. Examples of

using the Motivation Model in IS research are Fagan et al. (2008) in studying the intention to use computers and Cocosila et al. (2009) in investigating new technology acceptance. The Motivation Model has a low explanatory power of 38% (Venkatesh et al., 2003).

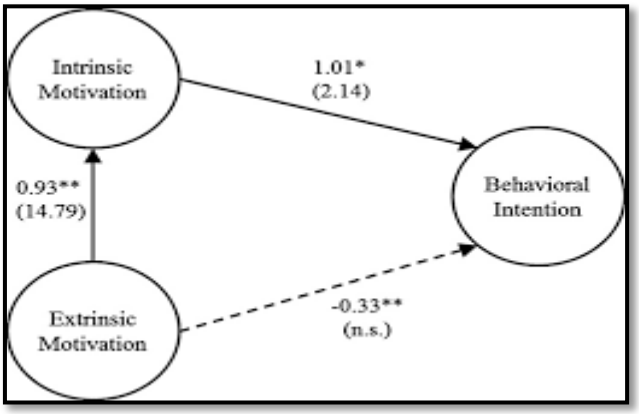

Figure 3.7: Motivational Model (Ryan & Deci, 2000)

3.3.7. Technology Acceptance Models

Davis (1985) introduced TAM with the aim of modelling users' acceptance of IS and technologies. This model was adapted from the theory of reasoned action to explain computer usage behaviour. The model uses the constructs PU and PEOU to explain an individual attitude towards using technology. This influence on attitude impacts the end user's intention to use the technology, leading to actual usage. PU is defined as the perception of users that using an information system or technology will increase their performance, and PEOU refers to how easy it is to use the information system or technology. Venkatesh and Davis (1996) suggested that there was no need for the attitude construct by demonstrating that PU and ease of use directly influenced behaviour intention. This resulted in the final version of TAM, which is illustrated in Figure 3.8.

TAM is one of the most widely cited models (Veiga et al., 2001). From its introduction in 1989 until 2007, there were over 700 citations of this model. TAM is still being applied to various domains to study technology acceptance, such as e-learning (Alfadda & Mahdi, 2021), autonomous vehicle adoption (Yuen et al., 2021), e-service quality (Ahmad et al., 2020), e-wallet (Farida & Ardiansyah, 2022), internet banking (Vuković et al., 2019).

Its strength is in its simplistic nature of determining intention to use, since intention to use is determined by only PU and PEOU (Bagozzi, 2007). However, researchers also view this strength as a limitation in situations where outside factors determine the intention to use. TAM

is too generalised to explain why users accept technology since PU and PEOU may be influenced by various external factors that influence the use of technology (Alshammari & Rosli, 2020; Bagozzi, 2007; Chuttur, 2009). When moderator variables such as gender and experience were added to the TAM model, an explanatory power of 52% was demonstrated (Venkatesh et al., 2003).

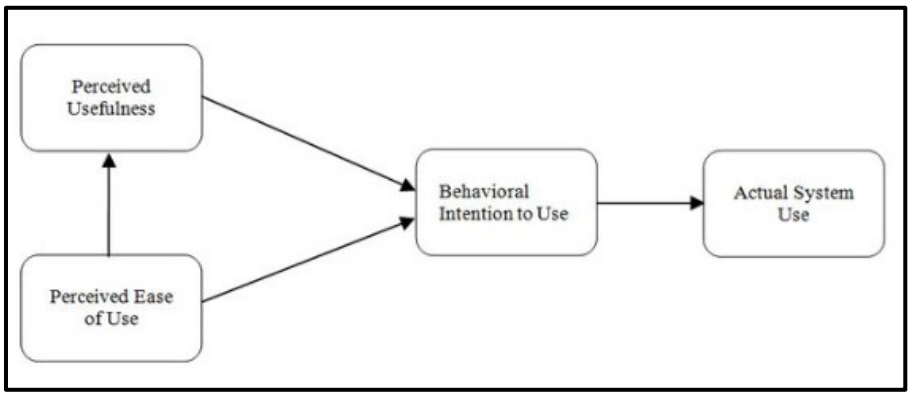

Figure 3.8: Technology Acceptance Model (Davis, 1985)

Venkatesh and Davis (2000) extended the TAM to TAM2 (Figure 3.9) by using social influence (SI) and cognitive processes to explain PU and BI to address TAM's criticisms. This model was validated in a longitudinal study spanning four organisations, with results indicating that both SI processes (subjective norm, voluntariness, and image) and cognitive instrumental processes (job relevance, output quality, result demonstrability, and PEOU) significantly influence user acceptance.

TAM2 was the chosen theoretical model in studies concerning Web2.0 usage behaviour (Wu et al., 2011), online learning management systems (Khoa et al., 2020), enterprise resource planning (Lubis et al., 2019), Internet of Things (Jain & Vijayakumar Bharathi, 2021), mobile banking (Milly et al., 2021), cloud computing (Purnama & Ginardi, 2019).

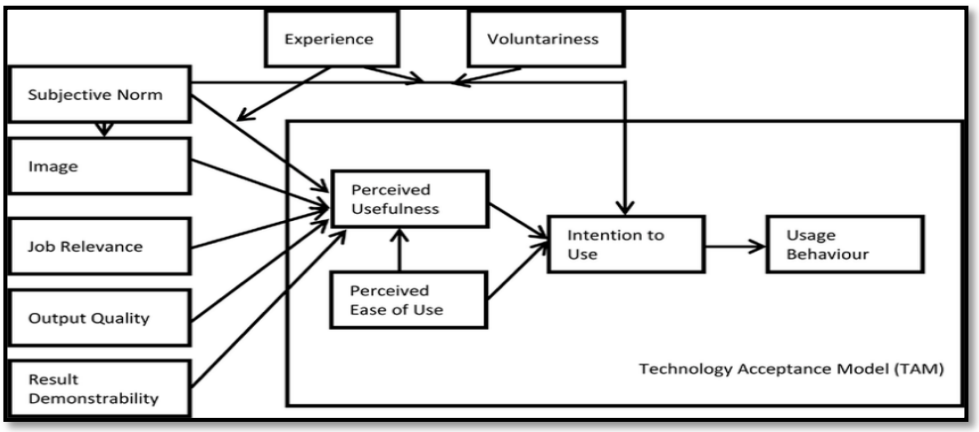

Figure 3.9: Technology Acceptance Model 2 (Venkatesh & Davis, 2000)

Venkatesh (2000) developed a model of the determinants of PEOU. The model illustrates that PEOU is determined by internal control (computer self-efficacy), external control (facilitating conditions), intrinsic motivation (computer playfulness) and emotion (computer anxiety) during the initial stages of the interaction of users with IT and as users become accustomed to the technology perceived enjoyment and object usability also become a determinant of PEOU. Combining TAM2, Venkatesh and Davis (2000) and the determinants of PEOU (Venkatesh, 2000), Venkatesh and Bala (2008) developed TAM3 (Figure 3.10). The different types of determinants indicated in TAM3 for the PU and PEOU constructs are individual differences, system characteristics, SI and facilitating conditions.

TAM3 was used to study the acceptance of e-government (Putra & Samopa, 2018), e-learning (Setiyani et al., 2021), cloud computing (Nikolopoulos & Likothanassis, 2018), electronic health records (Ebnehoseini et al., 2020).

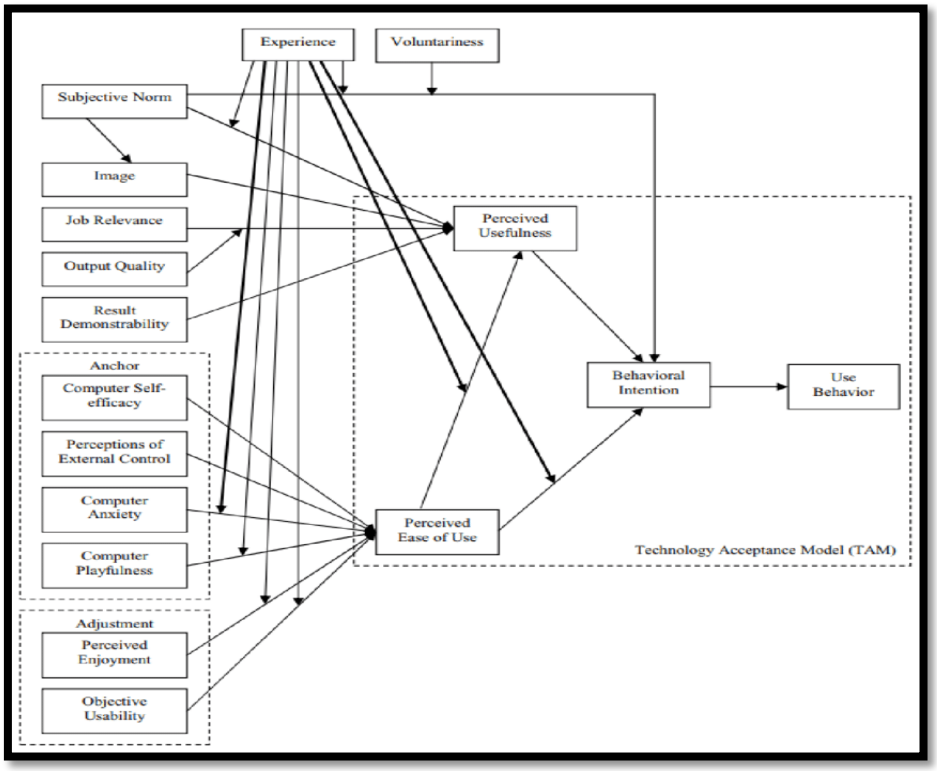

Figure 3.10: Technology Acceptance Model 3 (Venkatesh & Bala, 2008)

3.3.8. Combined TAM and Theory of Planned Behaviour (C-TAM TPB)

TAM is a predominant model measuring the factors that influence end-users' technology usage. However, Taylor and Todd (1995a) state that this model only focuses on acceptance of technology that has experience in a particular technology and does not focus on novice users. Studies have found that apart from the PU and PEOU factors significant in determining

acceptance of technology, social and control factors found in the TPB are just as important. According to Taylor and Todd (1995a), incorporating the models of TAM and TPB results in a more robust model that can determine the acceptance of technology by experienced and inexperienced users. In this model (Figure 3.11), usage behaviour is determined by BI and perceived behavioural control. Attitude, subjective norm and perceived behavioural control affect the end user's BI. Attitude towards a technology is determined by PU and ease of use. C-TAM-TPB was used in the study of internet banking (Safeena et al., 2013), Fleet Management Software (Tseng et al., 2013), e-learning adoption (Le et al., 2022), learning behaviour in Massive open online courses (Yang & Su, 2017).

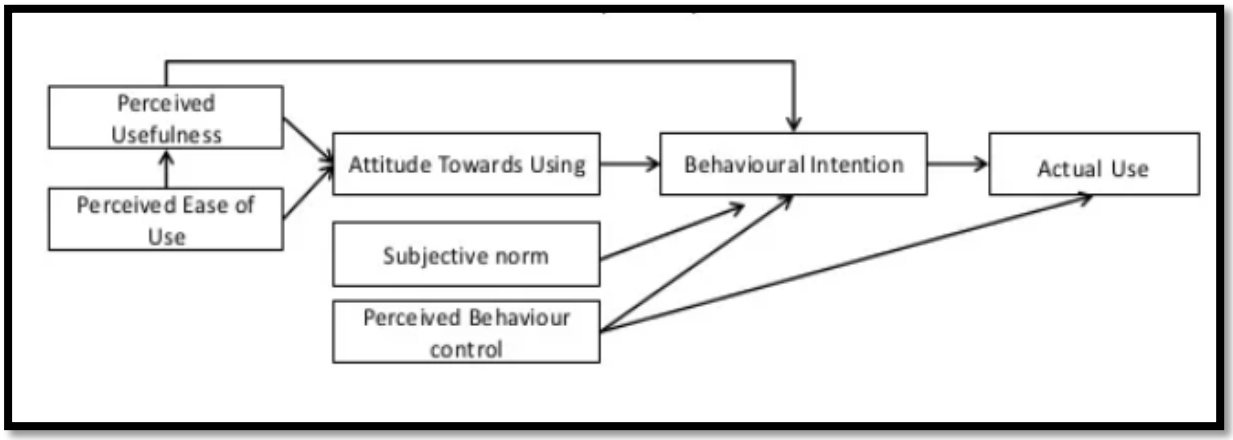

Figure 3.11: Combined TAM and Theory of Planned Behaviour (Taylor & Todd, 1995a)

3.3.9 UTAUT

According to Venkatesh et al. (2003), in the eight models discussed previously, the individual predictive powers of behaviour intentions to use technology ranged from 0.17 to 0.53 percent. Therefore, Venkatesh et al. (2003) aimed to develop a model to improve the predictive power of behavioural intentions to use technology. The constructs that determined acceptance were scattered in various models and by integrating their key constructs from these eight models and theoretical frameworks, Venkatesh et al. (2003) developed the unified theory of acceptance and use of technology known as UTAUT (Figure 3.12). The four principal determinants of intention and usage in the UTAUT model are performance expectancy (PE), effort expectancy (EE), social influence (SI) and facilitating conditions (FC).

Performance expectancy is *"*The degree to which an individual believes that using the system will help him or her attain gains in job performance*"* (Venkatesh et al., 2003, p. 447). This

construct was derived by using the PU in TAM, Extrinsic motivation in Motivation Model, Job-fit from (MPCU), Relative advantage from Innovation Diffusion Theory and Outcome expectations from the social cognitive theory.

Effort expectancy "is the degree of ease associated with the use of the system" (Venkatesh et al., 2003, p. 450). This construct formed from the PEOU from TAM and complexity in MPCU.

SI "is the degree to which an individual feels that it is important for others to believe he or she should use the new system" (Venkatesh et al., 2003, p. 451). This construct was established from subjective norms that exist in TRA, TAM2, and combined TAM-TPB, social factors from MPCU and image in DOI.

Facilitating conditions "is the degree to which an individual believes that organisational and technical infrastructure exists to support the use of the system" (Venkatesh et al., 2003, p. 453). This construct is derived from perceived behavioural control from TPB, facilitating conditions from MPCU and compatibility from DOI. These constructs and the variables behaviour intention and use of IT are moderated by gender, age, experience and voluntaries. The interplay of these variables is depicted in Figure 3.12. According to Sharma et al. (1981, p. 291) moderator variables can be defined as "one which systematically, modifies either the form and/or strength of the relationship between a predictor and a criterion variable".

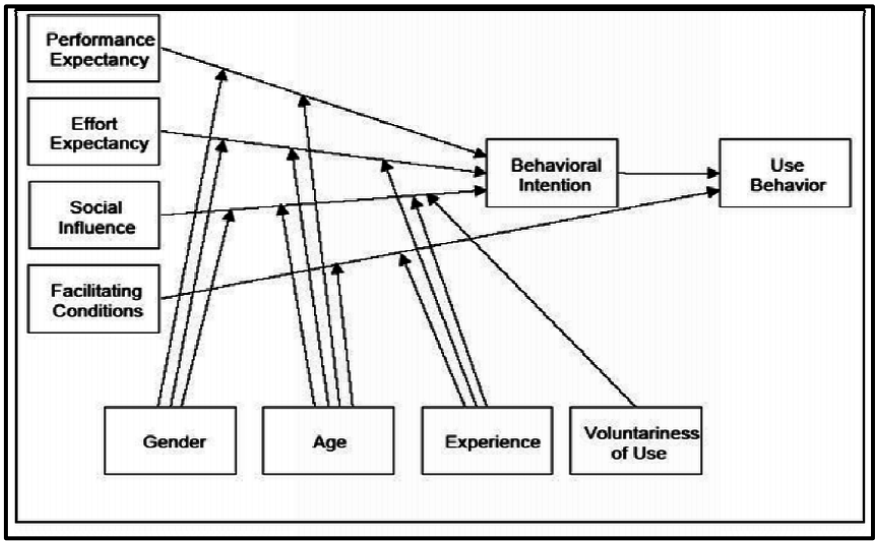

Figure 3.12: UTAUT (Venkatesh et al., 2003)

By 2014, the original article by Venkatesh et al. (2003) introducing UTAUT has been cited just under 5,000 times (Williams et al., 2015), with UTAUT being extensively used in different technological domains to study the adoption and acceptance of various technologies (Williams et al., 2015). Recently, UTAUT was used to study the use of social media in academic libraries

(Williams et al., 2021), mobile payment adoption (Wei et al., 2021), acceptance of e-government (Maznorbalia & Awalluddin, 2021), cloud computing adoption (Salam & Ali, 2020). A longitudinal study demonstrated the robustness of UTAUT in determining users' BI to adopt technologies (Venkatesh et al., 2003). This study found that the UTAUT model accounted for 70% of the predictive power in determining user intention to adopt technologies.

3.3.10. UTAUT2

Although UTAUT is very popular within an organisational setting, Venkatesh et al. (2012) extended this theory to include more constructs that can explain the usage of technologies by consumers. This was achieved by adding the constructs of hedonic motivation, price value and habit, resulting in the UTAUT2 theory, as illustrated in Figure 3.13. Hedonic motivation is the enjoyment gained from using technology. This enjoyment motivates the user to continue using the technology (Brown & Venkatesh, 2005; Davis et al., 1992; Venkatesh & Speier, 1999). Habit is defined as the "extent to which people tend to perform behaviours (use IS) automatically because of learning" (Limayem et al., 2007, p. 1). Price value is the price a consumer is willing to pay for the technology versus its benefits. Consumers will not pay for a technology that is too high and will question the quality of the technology if the price is too low (Dodds et al., 1991). UTAUT2 is a popular model with Tamilmani et al. (2021), indicating that it was cited over 6000 times in IS and other domains. Examples of recent UTAUT2 studies were used to investigate artificial intelligence (Gansser & Reich, 2021), autonomous delivery vehicles (Kapser & Abdelrahman, 2020), online gaming (Ramírez-Correa et al., 2019) and mobile banking (Kwateng et al., 2018).

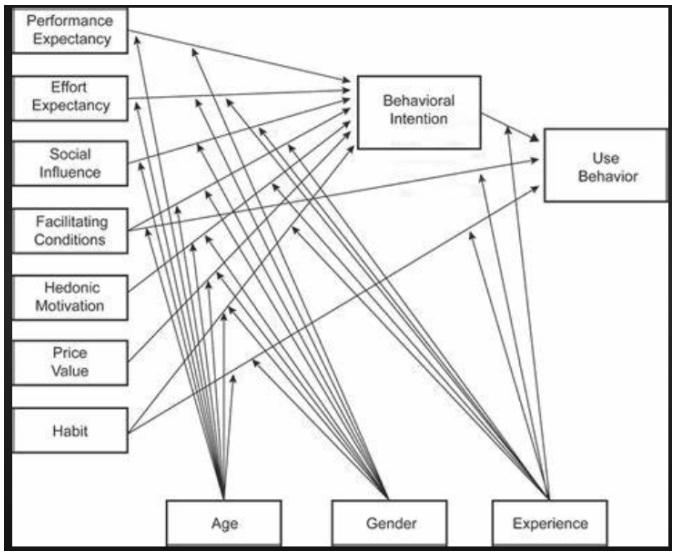

Figure 3.13: UTAUT2 (Venkatesh et al., 2012)

3.4. CRITICAL SUCCESS FACTORS

According to Rockart (1979), critical success factors (CSFs) are the limited areas within any business that demonstrate satisfactory results that will ensure the organisation's successful competitive performance. Things must go right within these areas for the business to succeed and grow. Bullen and Rockart (1981, p. 7) define CSFs as "the limited number of areas in which satisfactory results will ensure successful competitive performance for the individual, department or organisation. CSFs are the few key areas where 'things must go right' for the business to flourish and for managers goal to be attained". Similarly, Boynton and Zmud (1984, p. 1) describe "critical success factors as those few things that must go well to ensure success for a manager or an organisation and therefore they represent those managerial or enterprise areas that must be given special and high continual attention to bring about high performance." Hence, Boynton and Zmud (1984) define CSFs as the indispensable things that must be attained to yield the maximum competitive leverage. CSFs are essential in assisting organisations and project managers in achieving their project goals (Nasir & Sahibuddin, 2011). Furthermore, CSFs may vary according to the industry, domain and country (Yang et al., 2009). CSFs in DevOps for the present study can be defined as factors that enable projects deployed using the DevOps approach to meet or exceed their success criteria.

3.4.1 Success as the Dependent Variable

Time, budget, and meeting customer requirements are the three main criteria on which project success is based (Jones, 1995; Linberg, 1999; Pinto & Slevin, 1988). This is also commonly known as the iron triangle. However, project management success cannot only be measured by time, budget and specifications since important dimensions such as quality and project stakeholders' satisfaction also need consideration (Baccarini, 1999; Schwaber, 2004). Joosten et al. (2011) further suggest that success depends on multiple dimensions, but it is unclear which dimensions best represent the definition of success. The success of the iron triangle in measuring project success results from other factors that are not specific enough or measurable (Cuellar, 2010). Baccarini (1999) refers to success being measured using two categories: product success and project success. Product success involves measuring customer satisfaction and fulfilling organisational goals and objectives, while project success measures time, budget and specifications. According to PMI (2008), traditionally, cost, time and quality are essential components of project success from a project manager's viewpoint. Misra et al. (2009) are of the view that these three components are vital attributes that characterise the success of any

software development project, and most other success characteristics can be transformed into any of them. For example, productivity, customer satisfaction, and functionality can be perceived as quality criteria. Misra et al. (2009) further used the following five criteria to determine the success of Agile development projects: reduced delivery schedules, increased return on investment (ROI), increased ability to meet the current requirements, increased flexibility to meet changing customer requirements and improved business processes. According to Misra et al. (2009), these five criteria capture project success's three most important dimensions: reduced time, cost, and increased quality. Chow and Cao (2008), Stankovic et al. (2013) and Chiyangwa and Mnkandla (2017) also used quality, time and cost but also added scope (meeting all requirements and objectives) when studying CSFs in Agile software projects.

According to Forsgren et al. (2017), DevOps performance is determined along two main dimensions: throughput of code and stability of systems. Throughput is measured by how frequently a team can deploy code and how fast it can move from committing code to deploying it. Stability is measured by how quickly the system can recover from downtime and how many changes succeed versus how many fail.

3.4.2. Success Factors as Independent Variables

Numerous researchers have investigated the success factors in software development since its inception. The following discussion will focus on literature that recently studied these factors.

Chow and Cao (2008) conducted a quantitative exploratory study surveying 109 Agile projects and 408 respondents to determine CSFs in Agile software projects. It was a multi-national study targeting Agile Alliance users in the Americas, Europe, Asia/ Pacific, and Africa. Figure 3.14 shows the organisational, people, process, technical and project factors studied. Their results found that only 10 of their 48 hypotheses were supported, identifying Agile software engineering techniques, delivery strategy, and team capability as the three CSFs.

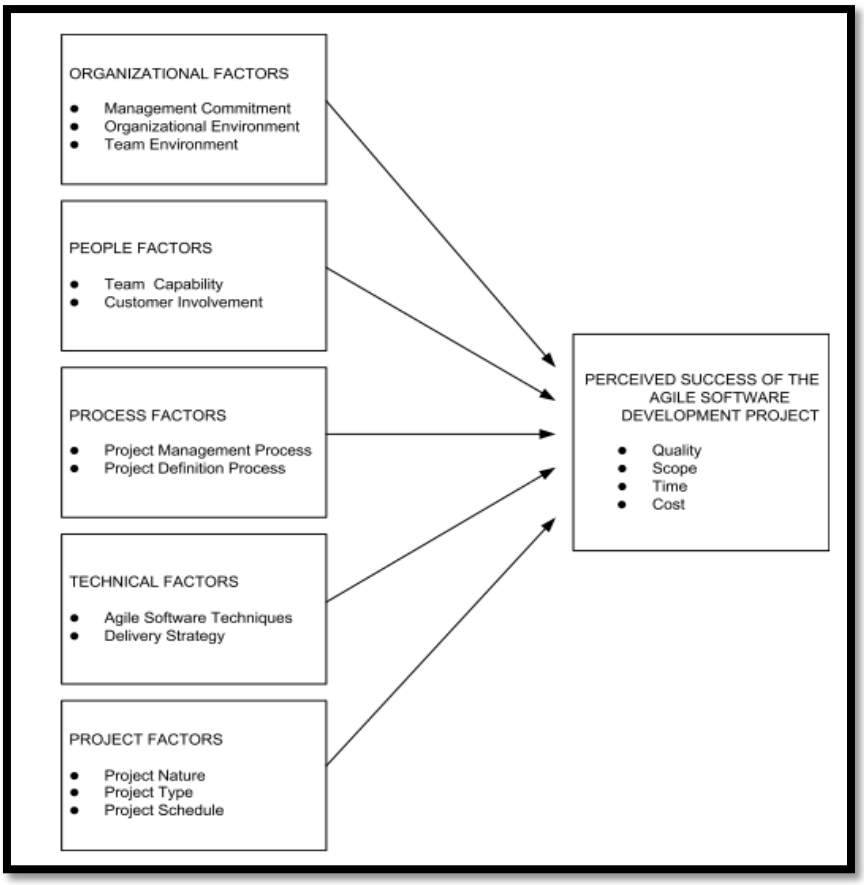

Figure 3.14: Factors affecting Agile success (Chow & Cao, 2008)

Misra et al. (2009) used a large-scale survey-based methodology to identify some CSFs in adopting Agile software development practices. Like Chow and Cao's study, respondents were from various industrial sectors across continents. However, this study only focused on people and organisational factors, as indicated in Figure 3.15. Of the 14 hypotheses tested, nine emerged to be significant with success. The significant hypotheses for the organisational factors were customer satisfaction, customer collaboration, customer commitment, decision time, corporate culture and control. For people factors, personal characteristics, societal culture, training and learning were significant.

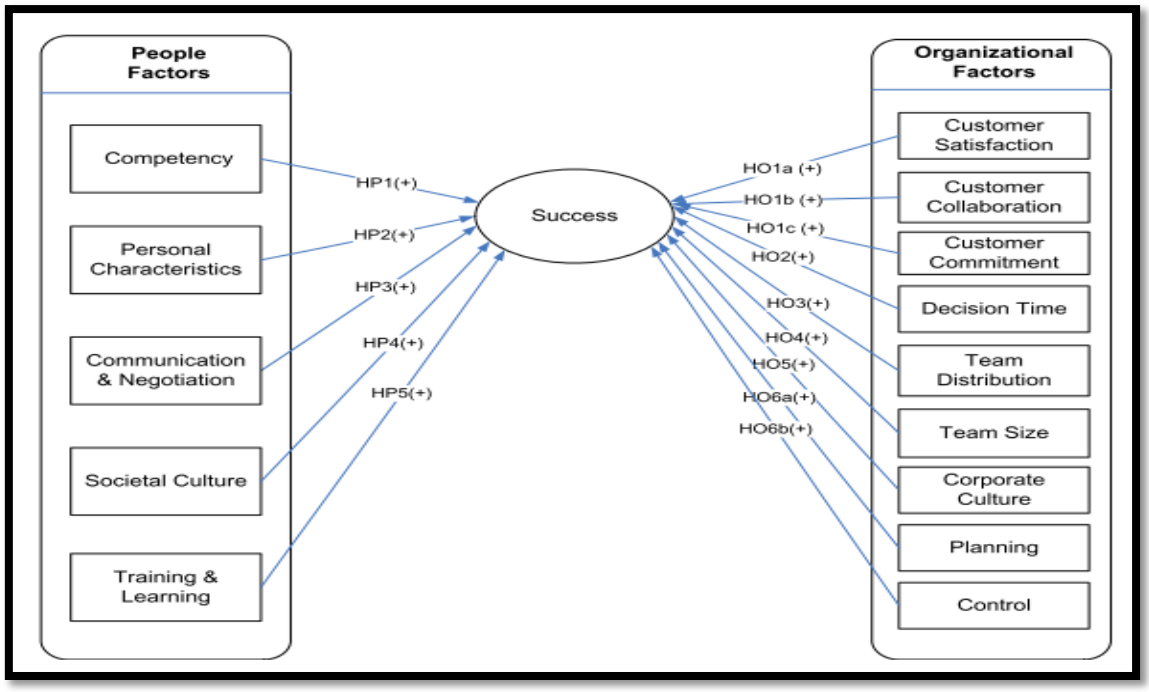

Figure 3.15: People and organisational factors affecting Agile success (Misra et al., 2009)

Mansor et al. (2011) presented a literature review of success determinants in Agile development methodology. Analysis of the existing literature points to customer involvement, communication, customer collaboration, corporate culture, simplicity, team members, code review, time allocation and minimum change of requirements as crucial determinants in Agile software development methodology success (Figure 3.16).

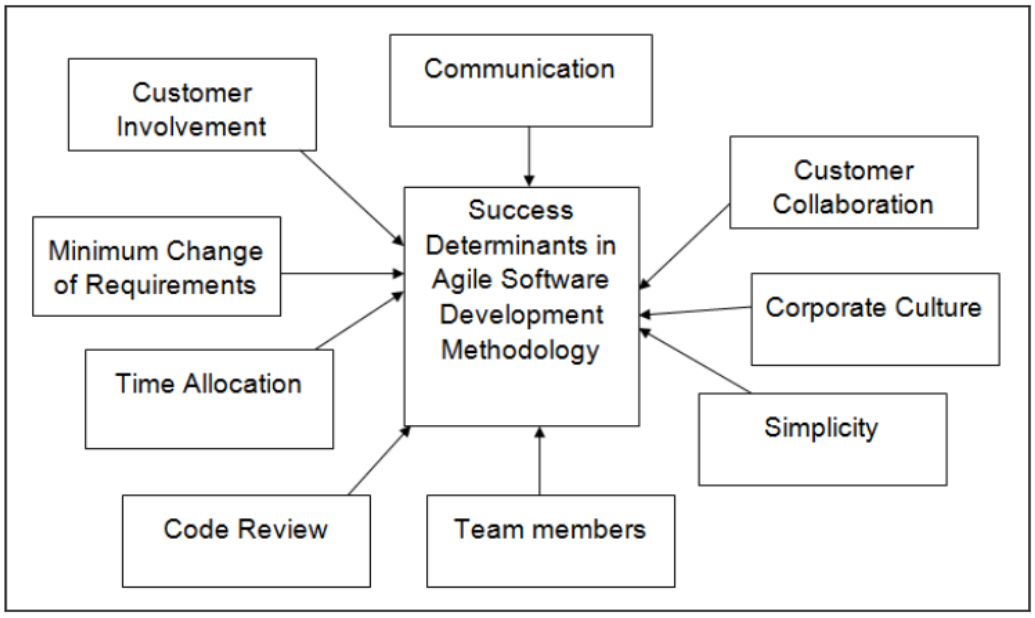

Figure 3.16: Critical determinants of Agile success (Mansor et al., 2011)

Sudhakar (2012) modelled CSFs for software projects from existing literature. The 35 CSFs were grouped into seven categories (communications, technical, organisational, environmental, product, team and project management). The identified CSFs include communication in the project, top management support, clear project goal, reliability of output, project planning, teamwork, project team coordination, quality control, client acceptance, the accuracy of output, reduced ambiguity, maximised stability, realistic expectations and user involvement. Based on the number of occurrences in the literature, communication in projects and top management support were the two most important factors in software development success. Figure 3.17 shows the CSFs identified and their relationship to project success.

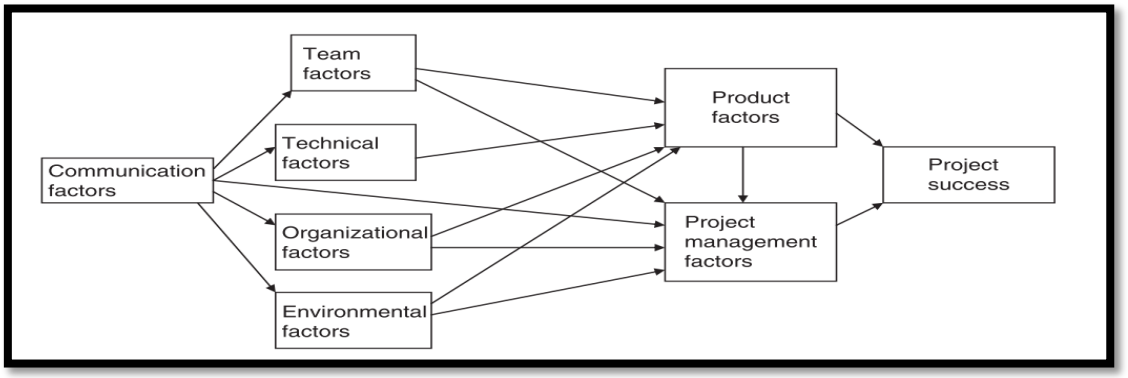

Figure 3.17: CSFs influencing project success (Sudhakar, 2012)

Nasehi (2013) used a case study approach to quantitatively determine CSFs in Agile software development projects. Regression analysis performed on the collected data has introduced ten factors that could be considered CSFs in terms of quality, scope, timeliness and Cost. Results match those from the previous study, suggesting that strong executive support and project type do not influence Agile projects' success.

Stankovic et al. (2013) investigated success factors in Agile software projects in former Yugoslavia companies. This study measured the same constructs as Chow and Cao (2008). Although the results introduced two more success factors in terms of timeliness and cost, this study could not confirm that either of the previous factors by Chow and Cao (2008) can be considered CSFs of former Yugoslavia IT companies.

Vithana et al. (2015) investigated the success factors of Agile software development in Sri Lanka by studying people and organisational factors. This study found that customer satisfaction, customer commitment, team size, corporate culture, technical competency, decision time, customer commitment and training and development influence the project's success.

Analysis of existing literature by Ramadan and Rizk (2015) classified success factors into five dimensions (organisational, people, process, project, and technical). In addition, the main factors with corresponding sub-factors were identified for each dimension. This results in a multidimensional success factor model, as illustrated in Table 3.1.

Table 3.1: Framework of success factors (Ramadan & Rizk, 2015)

Dimension	Main Success Factors	Sub Success Factors
Organizational	Corporate Culture	Support from top management
		Team Environment
People	User involvement	Handling commercial pressures
		Stakeholder politics
	Team Capability	Effective project management skills
		Ability to handle the project's complexity
		Decision time
		Effective communication and feedback
Process	Project management process	Minimum change in requirmeents
		Simplicity in process
		Good reporting of project status
	Project definition process	Risk management
		Time allocation
		Accurate estimates of project resources
	Active testing	Code review
Project	Clear objectives and goals	Project type
		Project nature
	Realistic schedule	
	Realistic budget	Team distribution
		Team size
	Clear requirements and specifications	
Technical	Selecting proper agile method	Configuring the necessary tools and infrastructure
	Using advanced technology	Familiarity with technology

A qualitative phenomenological study by Nguyen (2016b) was conducted to understand the lived experiences of 10 Agile software development project managers. Data from an open-ended email questionnaire shows that Agile software development teams need strong customer involvement, good Agile project management processes, product owner helps maximise the business value delivered by the team, sound Agile engineering techniques, and suitable technologies and development tools. These factors will influence the success of Agile software development projects.

Dikert et al. (2016) performed a systematic literature review on large-scale Agile transformations' challenges and success factors. Reviewing 53 publications, they grouped 29 success factors into eleven categories. The most noticeable success factors were management

support, choosing and customising the Agile model, training and coaching, and mindset and alignment.

Abd et al. (2016) identified and classified CSFs of Agile software development methodology using a mind map by conducting a systematic literature review. Their findings from previous research indicate that process, people, product, organisation, technical and project are key success dimensions. For the people dimension, Abd et al. (2016) mapped motivation, commitment, report and adapt, project champion, communication skills, management style, customer-centric issues and education and support as essential success factors. The success factors for the organisation are corporate culture, team distribution, organisational environment and sustain agility. High expertise, trouble shooting for team, practices, tool usage were important in the technical dimension. Literature for the process dimension points to regular delivery, proper method, simplicity, requirement gathering and integrate to external process as key factors. Team size, project type, schedule, pilot project and minimum change of requirement were success factors for the project dimension and software reuse and software characteristics in the product dimension (Figure 3.18).

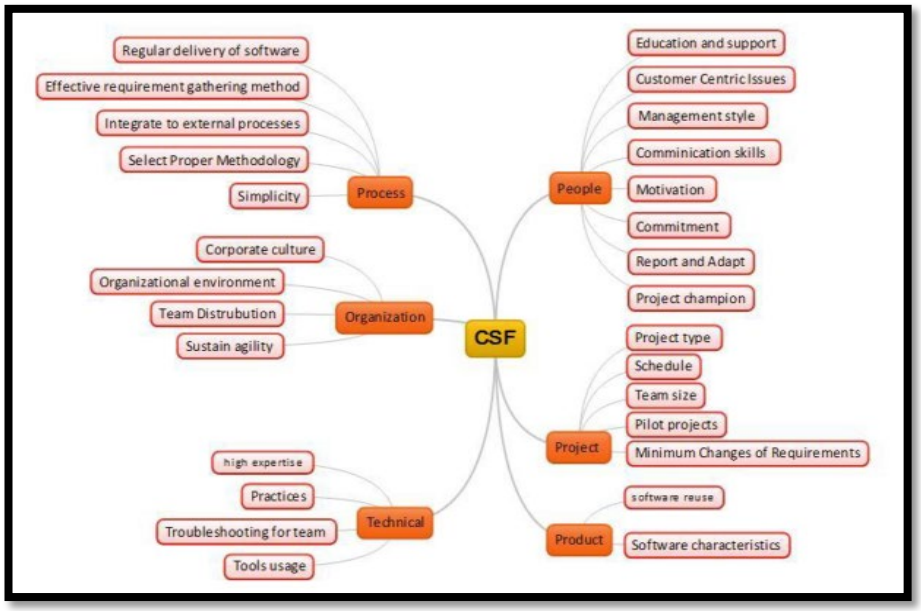

Figure 3.18: CSFs mind map model (Abd et al., 2016)

Yaghoobi (2017) prioritised key success factors of software products using a quantitative research approach that targets project managers, programmers/developers, designers, analysts and software testers. Pairwise comparisons using the fuzzy AHP method showed that efficient project management skills/methodologies, strong executive support, clear requirements and

specifications and team members with high competence and expertise are significant factors affecting a software development project.

Chiyangwa and Mnkandla (2017) modelled the success factors of Agile software development projects in South Africa. Structural equation modelling was used to study organisational, process, people, technological, and project dimensions against success and performance expectancy. Results indicate that performance expectancy and process factors have a causal relationship with actual success, and organisational, process, people, and project directly affect performance expectancy.

Aldahmash et al. (2017) analysed literature from 2006 to 2016 and identified eight factors as CSFs. These factors are delivery strategy, agile development techniques, organisational culture, communication, team capability and training, customer involvement, top management support and project management process. These factors were then mapped into the technical, organisational, people and process dimensions, as depicted in Figure 3.19.

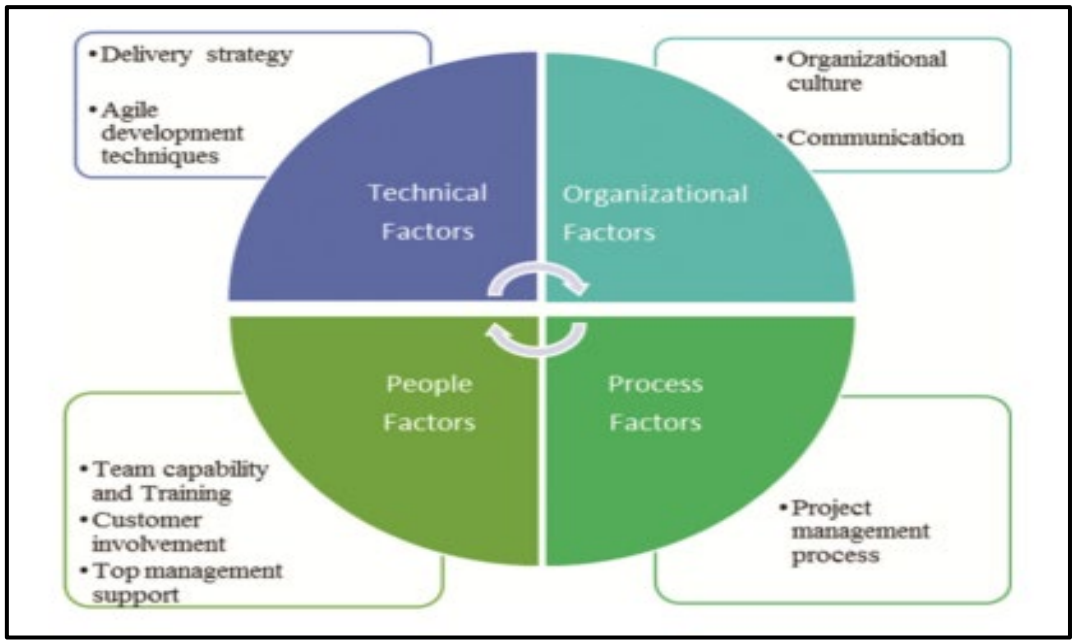

Figure 3.19: Agile development success factors taxonomy (Aldahmash et al., 2017)

The correlation of CSFs with successful software projects was empirically verified by Garousi et al. (2019) in Turkey. Organisational, team and customer factors were investigated, and the items for the team factor were the most critical. These items were team experience with the software development methodologies and the team's expertise with the task.

Tsoy and Staples (2020) compared success factors in successful and failed Agile projects. Using similar factors as Chow and Cao (2008), their study found that the more successful project was stronger in almost all success factors than the failed project.

Using analytic hierarchy process-based prioritisation, Shameem et al. (2020) investigated management, technology, human resources, coordination and methodology as success factors for scaling agile methods. The technology category was found to be the most important category in the success of scaling agile methods.

Muhammad et al. (2021) investigated factors crucial to Agile development success. Using a Delphi method, the "people" and "technical" dimensions were found to be the first and second most significant factors driving agile success.

Bogopa and Marnewick (2022) studied the CSFs in software development projects in South Africa. Using a quantitative approach, results showed that people, processes, and technical factors significantly achieved the project outcomes.

As discussed in the literature above (e.g., Chow & Cao, 2008; Ramadan & Rizk, 2015), CSFs for software development can be categorised into five dimensions. These are people, process, organisational, technical and project (Figure 3.20).

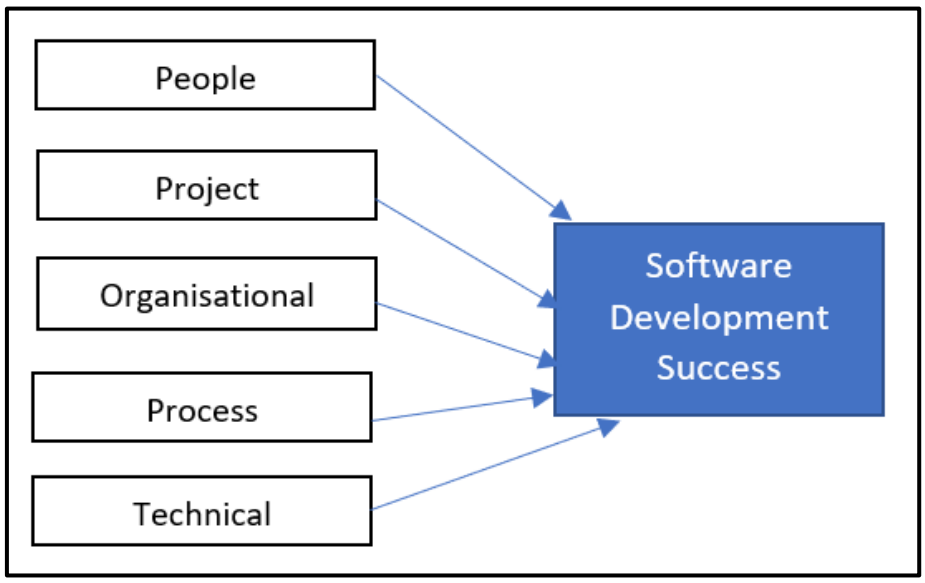

Figure 3.20: Major categories of Software Development Success Factors

3.5. CONCEPTUAL MODEL FOR THE STUDY

The main objective of this research is to model the acceptance and success factors for DevOps success. The literature outlined for acceptance factors shows that various theoretical frameworks exist to measure acceptance factors. The following discusses previous research

on the acceptance of software development methodologies. Khalifa and Verner (2000) examined the factors that influence the waterfall and prototyping model usage in software development. Findings showed that facilitating conditions and process quality were the key drivers resulting in the use of both models. By comparing five theoretical models (TAM, TAM2, Perceived Characteristics of Innovating (PCI), TPB, and MPCU) Riemenschneider et al. (2002) studied the user behaviour of a customised software development model. Usefulness, voluntariness and compatibility were significant determinants of user intention to use the development model. Hardgrave et al. (2003) studied the acceptance of an in-house development methodology using theories of intention formation and diffusion of innovation theory. Results indicate that developers' intentions are directly influenced by their perceptions of usefulness, social pressure, compatibility, and organisational mandate. Combining the theory of planned behaviour with the technology acceptance model, Hardgrave and Johnson (2003) examined the acceptance factors of a custom object-oriented software development model. They found that subjective norms, organisational usefulness, and personal ease of use play an integral role in the developers' intend to use the development model. Templeton and Byrd (2003) used a modified DOI theory to test whether developers' ease of use of an SDM significantly influences relative advantage, compatibility and trialability. The results were significant for compatibility, indicating that the SDM was compatible with the tasks developers needed to perform. Overhage and Schlauderer (2012) investigated the acceptance of Scrum using DOI theory. Results found that although Scrum provided relative advantages and was more compatible with developers' working habits, it added more complexity to the development process than traditional methodologies. Chiyangwa and Mnkandla (2017) studied Agile acceptance factors in South Africa, embedding UTAUT into the conceptual model. Performance and effort expectancy were found to be essential factors in Agile acceptance. Mudarikwa and Grace (2018), using the UTAUT, investigated the factors affecting Agile usage in the banking sector in South Africa. The result shows that facilitating conditions were the only significant factor affecting developers' usage of Agile in the banking sector.

Hence, there is limited research on acceptance factors in the domain of DevOps, which this study sought to determine using UTAUT2. As discussed in the topic "acceptance models", there are various acceptance models from which a researcher can select. Taylor and Todd (1995b) suggest choosing a parsimonious model with high explanatory power. However, Browne and Cudeck (1993) and McDonald and Marsh (1990) indicate that parsimonious models are only ideal if they can sufficiently explain the phenomenon under investigation and

therefore a parsimonious model is sacrificed for a more complex model in instances where it provides a greater understanding of the phenomenon. Although UTAUT2 is a less parsimonious model than other models (for example TAM), the number of factors included in this model makes it a better comprehensive model for determining acceptance. This is further illustrated by its high explanatory power of 74% for BI and 52% for technology use Venkatesh et al. (2016). A modified UTAUT2 model is used in this thesis study. The modification is the removal of the price construct since this model will measure factors affecting the acceptance of DevOps within an organisational context, where the DevOps professionals will not bear the monetary cost of implementing DevOps.

Most success factors discussed previously under the topic "Success Factors as independent variables" were within Agile environments. The Agile process is more focused on the development side. Since DevOps is a combination of development and operation, the current literature only provides a perspective from the developers, thereby leaving a gap in understanding from the perspective of other stakeholders, such as operations, security and testing. Also, there is a lack of academic literature on DevOps that addresses CSFs for successful DevOps projects. Although several studies (for example, Erich et al., 2014; Farroha & Farroha, 2014; Fitzgerald & Stol, 2015; Gupta et al., 2017; Kamuto & Langerman, 2017; Lwakatare et al., 2016; Perera et al., 2016) do mention automation and culture as important dimensions for success in DevOps, there is a lack of studies that quantitatively measure the success of these dimensions. Furthermore, there is a lack of research within the DevOps domain that incorporates all major software development critical factors under one model.

Erich et al. (2014) performed a mapping study on the cooperation between information system development and operations by analysing existing literature on DevOps. This analysis demonstrated that a culture of collaboration and automation are essential pillars in supporting DevOps. Furthermore, Farroha and Farroha (2014) and Fitzgerald and Stol (2015) opined that DevOps implementation entails a need to change the culture of teams. Farroha and Farroha (2014, p. 288) state that "*Once the new hires join the team, an environment needs to be created that is conducive to close collaboration. Hiring the right kind of people into the wrong culture will likely result in failure, just as having the wrong people in the right culture. New DevOps organisations need to change the culture and have the personnel in place that can make the customised changes permanent to meet emerging mission needs."* Gupta et al. (2017) studied the factors of Automation, Source Control, Cohesive Teams and Continuous Delivery by performing a quantitative study targeting DevOps professionals in 10 multinational companies.

Results derived from Structural Equation Modeling revealed that automation is a critical factor influencing Continuous Delivery. Lwakatare et al. (2016) used a qualitative study to examine important dimensions of DevOps success. Document Analysis and interview data indicated that collaboration, automation and culture are dominant factors. Perera et al. (2016) evaluated the impact of DevOps practice in Sri Lankan software development organisations using quantitative and qualitative methods. Culture and automation were found to be essential factors in the successful implementation of DevOps in Sri Lanka. Kamuto and Langerman (2017) used a qualitative approach to investigate the factors inhibiting the adoption of DevOps in large South African organisations. The dominant inhibitor was cultural change and suggested that organisations focus on team recognition, customer focus, performance feedback, innovation and engineering practices. Khan and Shameem (2020) studied challenging factors in DevOps adoption using the analytical hierarchy process (AHP). Results from this study indicate that people factors were a significant challenge to DevOps success. Akbar et al. (2020), using fuzzy AHP, further substantiated that the people factor is the most significant for DevOps success. Within a South African context, Rowse and Cohen (2021) carried out a quantitative study among DevOps professionals and found that culture and people factors in terms of team skills capability are important for DevOps success. Mishra and Otaiwi (2020) systematically mapped the current literature on DevOps software quality and found that automation and culture were the commonly cited factors in DevOps success due to its positive influence on software quality. A systematic literature review on CSFs of continuous practices in DevOps by Van Belzen et al. (2019) shows that organisational factors were widely documented as being critical to DevOps continuous practices success. A qualitative study of CSFs by interviewing software professionals found that technology, organisational and cultural factors are essential factors in DevOps success (Azad, 2022).

Hence, the literature on DevOps' success indicates that automation and culture are two additional factors that can lead to DevOps' success. Based on this, Figure 3.20 is modified to reflect the preliminary conceptual model used to determine DevOps acceptance and success factors (Figure 3.21).

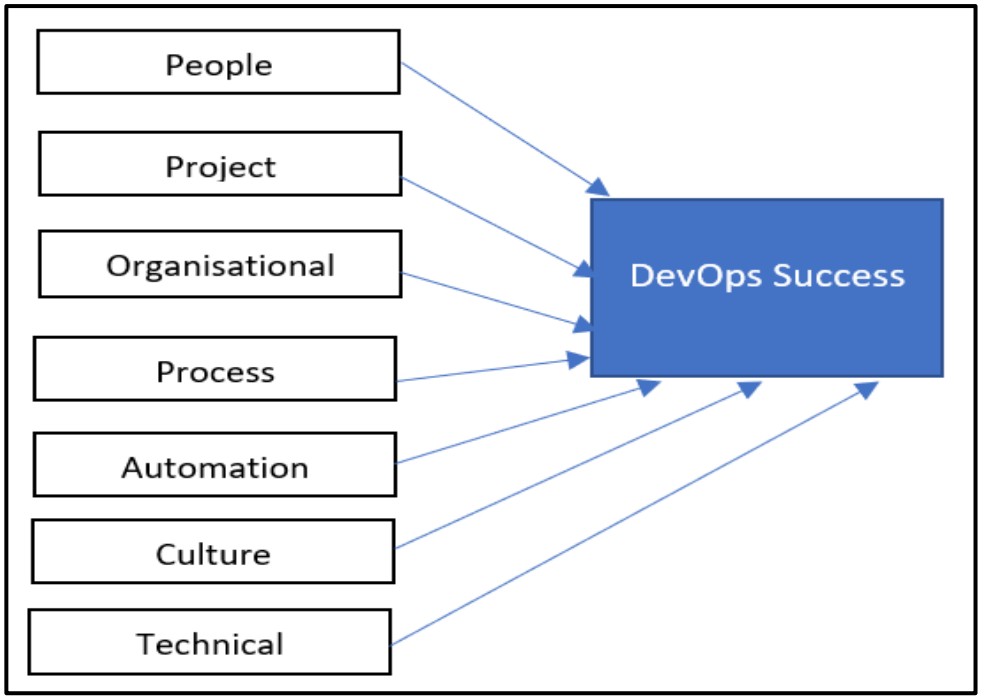

Figure 3.21: Factors affecting DevOps success

Based on the success factors, Figure 3.21, and the UTUAT 2 model, Figure 3.22 illustrates the initial conceptual model for the study. This model incorporates both the acceptance and success factors into a single model to provide an in-depth understanding of the factors that will drive DevOps adoption. According to this model, performance expectancy, effort expectancy, social, habit, hedonic motivation and facilitating conditions will positively drive BI to use DevOps. Intention to use DevOps with habit and facilitating conditions will lead to the actual usage of DevOps. Organisation, project, process, people, tools and technologies, product, automation and culture factors will result in the success of the DevOps project. The technical factor in Figure 3.20 and Figure 3.21 has been renamed "tools and technologies". This is in keeping with the DevOps principle of using tools and technologies. There is also a relationship between the actual usage of DevOps and DevOps success. The actual usage of DevOps can lead to more DevOps success. This preliminary model will lay the foundation to answer the research questions set out in this study. Initially, it will aid in answering RQ1. What are the factors affecting DevOps acceptance and success by being used as initial themes to aid the semi-structured interviews in the qualitative phase?

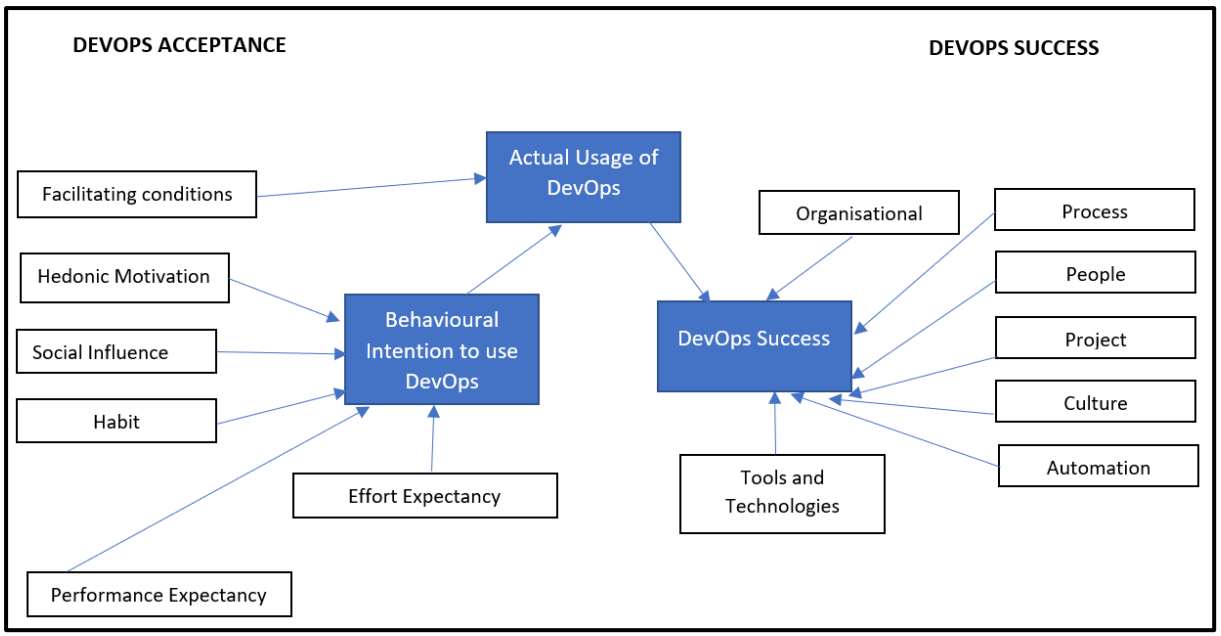

Figure 3.22: Preliminary Conceptual Model of Acceptance and Success Factors in DevOps

3.6 SUMMARY

This chapter aimed to provide an initial theoretical lens through which DevOps' acceptance and success factors can be researched. A review of the various acceptance models found that although the UTAUT2 model had the least parsimony, it had the highest explanatory power compared to the other acceptance models. Furthermore, UTAUT2 has key constructs of the other models embedded within it, making it a comprehensive model to use when studying acceptance.

A literature review on success factors indicated that most factors were studied in an agile context. Hence, there is a gap in understanding these factors within the DevOps context. Although automation, culture and organisational factors are documented in DevOps literature, there is still a lack of evidence indicating how important they are to the success of DevOps projects when studied together with other factors. The literature thus reaffirms the lack of research in DevOps concerning acceptance and success factors, especially in a South African context.

The initial conceptual model will be validated and extended based on the qualitative analysis of respondent interviews presented in Chapter 5. The next chapter will focus on research design and methodology, concentrating on the reasons for selecting various research techniques. This will ensure that the research questions are answered while maintaining the validity and reliability of this study.

CHAPTER 4: RESEARCH DESIGN AND METHODOLOGY

4.1. INTRODUCTION

The previous chapter discussed the theoretical framework for the study, highlighting the acceptance and success factors that may influence the success of DevOps. The current chapter provides the methodological blueprint to answer the research questions laid out for this study. Research design and methodology are important to the success of any research project. Scientific research practice is based on a well-planned research design and methodology strategy (Babbie & Mouton, 2008). A research design must be aligned with the philosophical assumptions and the preferred paradigm the researcher has adopted for the study (Creswell, 2013). The current chapter discusses the research design choice and the methodology used in the study's design. This chosen design and methodology will enable the researcher to answer the following research questions successfully:

1. What are South African software practitioners' perspectives on the DevOps phenomenon?
2. What acceptance factors influence software professionals' intention to use DevOps?
3. What success factors influence DevOps project success?
4. Which acceptance and success factors are critical to the adoption of DevOps?

4.2. PHILOSOPHICAL WORLDVIEW

Guba (1990) defines worldview as a set of beliefs and principles that guides a person's behaviour. A worldview is also commonly referred to as paradigms by Guba and Lincoln (1994). Creswell (2009) states that worldview is the "general orientation about the world and the nature of the research that the researcher holds". The researcher's discipline shapes this orientation, the beliefs of his peers, and the researcher's previous experiences. Candy (1989) suggests that research worldviews can be grouped into positivist, interpretivist, and critical. Tashakkori and Teddlie (2003) suggest a fourth type, the pragmatic paradigm. Creswell (2009) reclassified these worldviews as postpositivist, constructivism, advocacy/participatory and pragmatism. These worldviews will be discussed from an overview perspective to contextualise the researcher's personal worldview preference in the current study.

4.2.1. The Postpositivist Worldview

The positivist worldview supports the belief that there is unquestionable truth to scientific knowledge (Guba & Lincoln, 1994). Myers (2004, p. 1) states that "*Positivists generally*

assume that reality is objectively given and can be described by measurable properties, which are independent of the observer (researcher) and his or her instruments". The positivists believe that absolute truth can be found and that proof established in research is always perfect and fallible. However, many researchers refute this belief, resulting in the ***postpositivist*** worldview with the fundamental assumption that absolute truth cannot be determined in isolation and has to be qualified according to the role that this knowledge will play in a social context. Postpositivism is underpinned by the quantitative research approach, where a researcher investigates a phenomenon within a social context and the researcher's worldview may influence the study's outcome. The postpositivist worldview follows a scientific method of identifying a problem, deriving hypotheses, using an underlying theoretical framework and testing the theory by subjecting the hypotheses to empirical validation (Creswell, 2009).

4.2.2. Constructivism

Tashakkori and Teddlie (2003) state that constructivism has several names, such as interpretivism and naturalism. Constructivism is based on individuals' desire to understand the world in which they live and work. In seeking this understanding, subjective interpretations of objects or things are gained, resulting in multiple and varied meanings. This allows a researcher to acquire a complexity of opinions instead of reducing meaning to a few ideas (Creswell et al., 2003). Constructivists follow an inductive path that focuses on the generation of theory rather than the testing of the theory (Guba & Lincoln, 1994). To enable theory generation, the researcher follows a qualitative approach whereby open-ended questions are put forward to respondents regarding their experiences and interpretations of world phenomena.

4.2.3. Advocacy/Participatory

According to Creswell (2009, p. 9), the advocacy/participatory worldview "*holds that research inquiry needs to be intertwined with politics and a political agenda*" that addresses important social matters such as "empowerment, inequality, oppression, domination, suppression, and alienation". Hence, this worldview focuses on marginalised groups or individuals within society. Feminist perspective, critical theory, and queer theory are some of the common theoretical lenses used in studies that follow the advocacy/participatory worldview.

4.2.4. Pragmatism

This paradigm resulted from philosophers arguing that it is impossible to gain the truth about the real world by using just one scientific method, as indicated in the postpositivist paradigm or determine social reality by the interpretivist paradigm. According to Rossman and Wilson (1985), pragmatism advocates that a researcher needs to emphasising the research problem and resort to the use of a multitude of available approaches to understand and investigate the problem from perspectives of scale and depth-orientation. Hence, the pragmatic paradigm uses mixed methods to gain an understanding of human behaviour. The pragmatic worldview oscillates between induction and deduction, referred to as abductive reasoning, which occurs when inductive results obtained from a qualitative study can be regarded as inputs into a deductive quantitative study and vice-versa (Morgan, 2007). Hence, the pragmatic worldview is helpful for studies with objectives that need to be addressed using the subjective (constructivist) and the objective (postpositivist) paradigms.

In order to contribute to the adoption of DevOps in South Africa as well as to enrich the existing literature on DevOps acceptance and success factors, this study adopted a pragmatist position. This strategy has been adopted because of the researcher's epistemological stance that DevOps is best understood by acquiring knowledge of the *lived experiences* of software professionals in the context of DevOps usage in South Africa. This strategy will mitigate the dearth of literary content on this topic due to its recent implementation in the industry. Furthermore, the majority of the current literature documents DevOps from a practitioner's viewpoint that is mostly anecdotal and lacks the rigour and validity of an academic study. Also, studies documenting the acceptance and success factors are comprehensively documented from an Agile perspective, not a DevOps perspective. Therefore, in the context of the current study, an understanding of the DevOps phenomenon was first obtained inductively, and after that, important factors were deductively identified within the context of acceptance and success of the methodology in a professional organisational environment. This interplay between deduction and induction necessitated the invocation of a ***pragmatic approach*** that will guide this study from inception to conclusion.

4.3. RESEARCH APPROACHES

According to Saunders (2011), there are two types of research approaches: deduction and induction. Deduction is the approach commonly associated with scientific research and is common in natural sciences, where scientific laws explain phenomena that predict their occurrences (Collis & Hussey, 2003). Robson (2002) mentions there are five stages in deductive research: hypothesis deduction based on existing theory, hypothesis expressed in operational terms, operational hypothesis testing, examination of the outcome, and modification of the theory based on the outcome. Induction is an approach to understanding the nature of the investigated problem better with the objective of formulating theory. This is done by making sense of collected interview data that results in a theory formulation. This approach emphasises understanding the meaning humans attach to events (Saunders, 2011). The current study invokes both approaches. Induction is the initial approach whereby the researcher wants to gain an understanding from stakeholders in the DevOps teams about DevOps and its success. This is the qualitative phase of the study and will culminate in the identification of a tentative theoretical model for the study. Subsequent to this phase, deduction was used to test this theory via the strategy of proposing a set of hypotheses that have been subjected to empirical validation. The data used to validate the set of hypotheses was obtained quantitatively in the context of acceptance and success factors of DevOps usage in South Africa. Saunders (2011) supports this strategy by confirming that both approaches can be used in a study at different stages. The various phases of the study's research design will be presented and discussed in the narrative that follows.

4.4. RESEARCH METHODS

The main methods of research are presented as a backdrop to the discussion of the study's overall research design that is presented in Section 4.5. The reasoning behind this approach is that the discussion on research design makes liberal use of references to research methods and a prior discussion of these methods will assist in enabling an easier cognitive assimilation of the content presented in Section 4.5.

4.4.1. The Qualitative Method

Qualitative research seeks to understand phenomena in context-specific settings, such as "real-world setting [where] the researcher does not attempt to manipulate the phenomenon of

interest" (Patton, 2002, p. 36). This research approach is used in the study of complex situations to provide an in-depth and holistic analysis of a particular context from the point of view of the research participants (Blumberg et al., 2005). A small sample is usually used to collect data (Leedy & Ormrod, 2005). Qualitative studies typically generate theory using an inductive approach within a constructivist paradigm (Creswell, 2009). The most common research strategies followed in qualitative research are ethnography, grounded theory, case studies, phenomenological research, and narrative research. Ethnography is a strategy used when research is carried out on an intact cultural group in their natural setting. Data collected during ethnography can take the form of observational and interviews over a prolonged period. Grounded theory is a strategy used for developing theories. This is achieved by gathering and analysing data in multiple stages resulting in the creation of interrelationships between concepts and categories. Case study is an in-depth exploration of an event, program, activity and process using different data collection procedures (Merriam, 2002). Phenomenological research uses the lived experience of individuals to establish an in-depth understanding of a phenomenon. Using the concept of bracketing, the researcher sets aside his or her own experience in order to understand those of the participants. Narrative research is an approach that examines the existence of individuals by asking them to narrate stories about their lives (Merriam, 2002).

4.4.2. The Quantitative Method

Quantitative research uses deductive reasoning to test theories. This research is situated within the postpositivist paradigm. Research results are presented in the form of numerical information and is used to form relationships among measured variables (Leedy & Ormrod, 2005). This information can be collected quickly from many respondents and then converted into numerical indices (Leedy & Ormrod, 2005). Quantitative studies can involve intricate experiments with multiple variables and treatments. Complex structural equation modelling that indicates casual paths and the relationship between measured variables is commonly used to document the results of quantitative studies. Quantitative strategies can take the form of experimental designs and non-experimental designs such as surveys. Experimental design research strictly follows a scientific research design where experiments are executed in a controlled environment. During these experiments, the researcher collects data that will either support or reject hypotheses that were identified at the inception stage of the study. Survey research is used in studies that want to measure a population's attitudes, trends or opinions. These studies can be cross-sectional or longitudinal, using questionnaires or structured

interviews for data collection. Survey research aims to generalise findings from a sample to a population (Maree, 2007). These studies are generally deductive and invoke existing theory as a guiding framework.

4.4.3. Mixed Methods

Mixed method research combines both qualitative and quantitative approaches during research inquiry. While the main methods entail qualitative and quantitative inquiry, a mixed methods study resonates strongly with the researcher's worldview perspective and typically entails mixing both approaches into a single research study, thereby enhancing the overall strength of the study by increasing the breadth and depth of understanding (Creswell & Clark, 2007; Johnson et al., 2007; Wisdom et al., 2012). Creswell and Clark (2007) further suggest that mixed methods research is appropriate when a single source does not provide sufficient data to meet the research objectives, there is a need for further explanation of results, findings from exploratory study need to be generalised, or when research objectives are too complex and need to be tackled using different phases or various types of data. Since mixed methods research combines qualitative and quantitative approaches, the sequence of these approaches in a research study will dictate the type of mixed research approach. The two broad categories are concurrent design and sequential design. In concurrent design, the strategy is to simultaneously implement the qualitative and quantitative phases of the study, and the results are compared. In a sequential design, there are two options that may be invoked. These are the explanatory and exploratory sequential research design options. The explanatory designs implement the quantitative approach first and then deploy a qualitative approach to better understand the results of the quantitative phase. The exploratory designs first implement a qualitative phase, and the results will be used to generate hypotheses or research instruments. A quantitative phase is then deployed to test the hypotheses or measure the variables in the research instrument. Depending on the study's research objectives, the qualitative and quantitative phases in the sequential design may be equal, or the weighting of either can be higher (Creswell, 2009).

4.5. RESEARCH DESIGN ADOPTED FOR THE CURRENT STUDY

4.5.1. Sequential Exploratory Research Design

This study adopted a **sequential exploratory mixed methods** approach consisting of two phases. The first phase consists of qualitative data collection and analysis, followed by a quantitative data collection phase. Since DevOps is a new phenomenon within software development, a qualitative approach was first used to study this phenomenon by interviewing software professionals who have adopted DevOps. The findings from this phase were reconciled with content from the study's literature component (Chapter 2) to develop an operational model subjected to quantitative inquiry. The developed model has been aligned to the study's main research questions so that these questions could be subjected to empirical validation, discussion and conclusion. The sequence of phases that have been adopted in the current study is illustrated in Figure 4.1. The research methods that have been employed for each of these phases are presented in the discussion that follows the illustration in Figure 4.1.

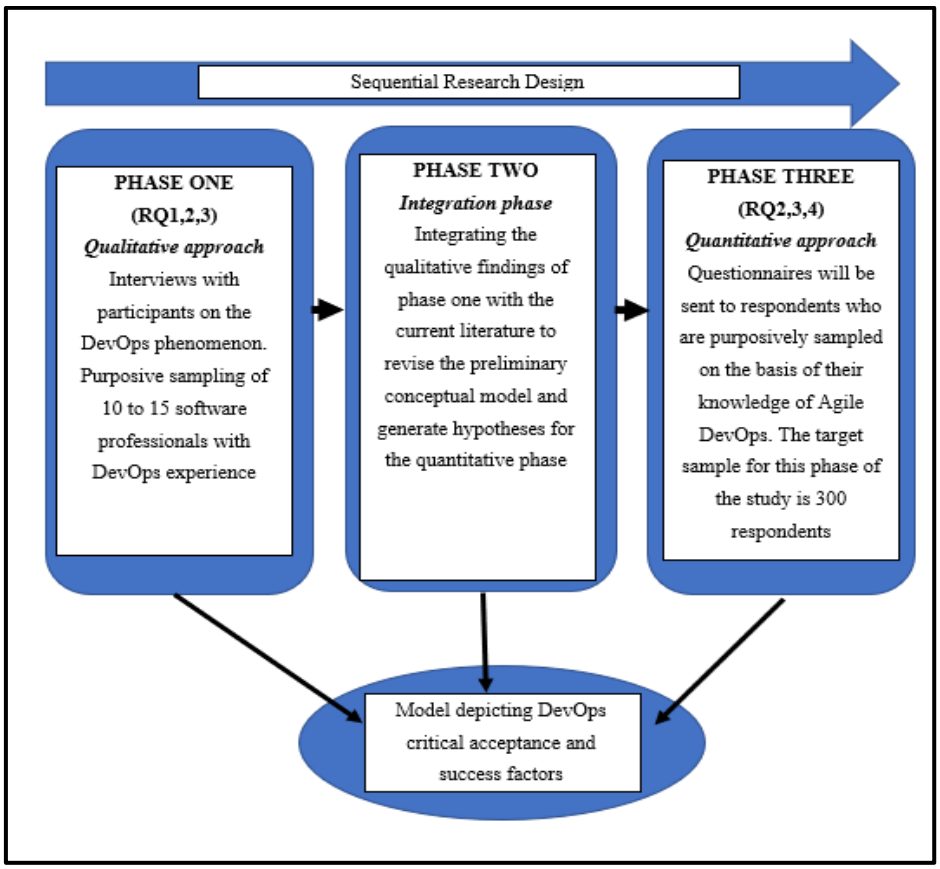

Figure 4.1: The study's sequential Research design

4.5.2. The Qualitative Phase

This phase enabled the researcher to obtain insights into the lived experiences of various stakeholders regarding DevOps. According to Creswell (2012), studies that are interested in obtaining the lived experiences of individuals regarding a concept or phenomenon are referred to as phenomenological studies. The results from this qualitative phase were used to validate and revise the preliminary conceptual model derived from the literature. A quantitative phase was then undertaken to collect and analyse empirical data from DevOps professionals to validate the revised conceptual model empirically.

This phenomenological process starts with identifying a phenomenon of human interest and then eliciting details of people's experiences with this phenomenon by virtue of their interactions with it. The intention is to create a correlation between how stakeholders experienced the phenomenon and what value may be derived from the experience with this phenomenon (Groenewald, 2004)

There are two main types of phenomenology paradigms. These are descriptive (also referred to as Transcendental) and interpretive (Heuamantics). Edmund Husserl developed descriptive phenomenology to deliver a philosophy different from objectivity and positivism in natural sciences. He believed that, unlike natural sciences, where phenomena can be studied and explained by experiments in an environment whereby certain variables are controlled not to influence the phenomenon, phenomenology establishes the lived experience of respondents towards a phenomenon by accessing their inner subjectivity. Husserl further believes that in order for a respondent's perception of a phenomenon to reflect their owned lived reality, it is essential that the researcher removes any biases of the phenomenon, thus portraying an in-depth description of the phenomenon from the respondent's perspective only ensuring that the respondent is integral to the understanding of the phenomenon. Husserl proposed a strategy whereby the researcher removes any preconceived biases of the phenomenon and isolates any existing knowledge of the phenomenon during the study's data collection and data analysis phases and referred to this strategy as transcendental subjectivity (Lopez & Willis, 2004). Transcendental subjectivity is achieved by a process called bracketing. Van Manen (1984) explains bracketing as removing any prior knowledge from the study that the researcher has or exists in the literature. However, Martin Heidegger rejected Husserl's viewpoint of bracketing, stating that it is important to go beyond just describing a phenomenon to trying to understand the meanings of these experiences in the context of everyday occurrences. For this to be achieved, it will be difficult for a researcher to interpret the meaning of respondents without

allowing their pre-existing knowledge and biases to influence their interpretation of the respondent's experience. This paradigm of phenomenology is called interpretive phenomenology (Kafle, 2013; Reiners, 2012). In the context of the current study and the researcher's worldview perspective, an interpretive phenomenology approach was used in this study. Within the backdrop of interpretive phenomenology, the sampling and data collection phases adopted for the study are presented and discussed in the following sections.

Study Site and Target Population

The research was conducted with software professionals in South Africa. Data was gathered from various stakeholders within the DevOps community (DevOps team leaders, operations, software development, and quality assurance personnel). The study's respondents were chosen purposively as the target population because of their experiential knowledge of DevOps, rendering this group the most appropriate population for the study.

The Qualitative Sampling Strategy

Convenience, purposive, and theoretical sampling are the three methods used in qualitative research (Lopez & Whitehead, 2013). Convenience sampling is the most cost-effective method, allowing participants to be targeted in less time and effort since it enables the recruitment of participants that are easily accessible and convenient to the researcher. Since researchers are free to select participants based on convenience, the findings may be viewed as poor quality and unreliable (Lopez & Whitehead, 2013). Theoretical sampling is commonly used in grounded theory research, allowing a researcher to select a new sample after analysing data obtained from an initial sample. Hence, the researcher can repeatedly sample to develop a theory and, therefore, select a new sample to test the validity of the theory (Lopez & Whitehead, 2013). According to Leedy and Ormrod (2005) and Remler & Van Ryzi (2011), purposive sampling is the most common method in qualitative studies. Purposive sampling, also known as judgement sampling, is used when the cases are selected based on the researcher's judgement (Saunders, 2011). This involves the researcher having prior knowledge of the research to identify respondents who will provide the best information for the study based on experience, status or specialised knowledge (Lopez & Whitehead, 2013; Sekaran & Bougie, 2010).

Purposive sampling was used for the qualitative part of this research. The researcher used this sampling approach to identify individuals within the DevOps community who were able to provide specialised knowledge on the DevOps phenomenon.

The Sample Size

Englander (2012) believes there is no definite answer to the correct sample size in qualitative research. Researchers believe that qualitative research is not the amount of data collected but rather the richness and depth of data collected. This will allow for a greater understanding of the issues being investigated, enabling the researcher to attain the study's objectives. However, many researchers provide sampling guidelines for qualitative studies, usually based on the type of qualitative research that is employed. Creswell (2013, p. 239) suggests the following guidelines regarding the sample size for qualitative research:

- In narrative research, a sample size of two would be adequate.
- In phenomenological research, a sample size in the range from 3 to 10 is advocated
- In grounded theory research, a sample size of 10 to 30 is advocated
- In case study research, there should be a study of at least four to five cases.

Since the current study used a phenomenological approach, at least 10 participants were targeted. Mason (2010) and Fusch and Ness (2015) views on data saturation were also considered during the interview process. Data saturation is when new data tends to be redundant to data already collected. Data saturation is reached in interviews when the researcher hears the same comments repeatedly. This becomes a signal to the researcher that no new information is being generated, and it is time to stop collecting information and start analysing what has been collected.

The Qualitative Data Collection Strategy

For the qualitative phase of this study, data was collected using interviews. Interviews can be categorised as structured, semi-structured and unstructured. This study used semi-structured interviews since the researcher identified initial themes from the literature that the participants needed to elaborate on. This allows the interview to remain focused but permits the researcher to explore other relevant ideas that may come up during the interview, which can further enhance the understanding of the phenomenon being investigated (Adeoye-Olatunde & Olenik, 2021). Hence, this interview technique allowed the researcher to ask additional questions. The researcher identified 20 DevOps experts in the industry through industry collaboration and via

the alumni of the University. These individuals were invited to participate in the research, to which only nine agreed. A date, time and place were then set up for the interviews. All interviews were conducted face-to-face at the participants' employment and recorded. The researcher also kept a journal to make points that needed clarification later on. The interview schedule is documented in Appendix A.

Qualitative Data Analysis

This study's qualitative phase involved using qualitative data analysis techniques. All recorded interviews were transcribed and then verified to ensure accuracy with the recorded interviews. All transcriptions were then uploaded to the qualitative data analysis software Nvivo. The phenomenological transcripts were subjected to content analysis since it was identified as the most appropriate technique. According to Zhang and Wildemuth (2005), the primary objective of qualitative content analysis is to reduce data to themes or categories and to document the detailed descriptions of the lived experience of participants with the phenomena. Zhang and Wildemuth (2005, p. 11) further state that "Through careful data preparation, coding, and interpretation, the results of qualitative content analysis can support the development of new theories and models, as well as validating existing theories and providing thick descriptions of particular settings or phenomena". If there is an existing theory to validate and extend, as is the case of the current study, Hsieh and Shannon (2005) refer to this approach as *directed content analysis*. Content analysis provides two approaches to data analysis: conceptual and relational analysis. Conceptual analysis looks for the frequency of themes in a text, while relational analysis examines relationships among concepts in a text (Wilson, 2016). The transcribed data was subjected to both conceptual analysis, resulting in the generation of themes, and relational analysis, which resulted in the creation of new relationships. This analysis was facilitated by the software package Nvivo.

Trustworthiness of Qualitative Research

According to Lincoln and Guba (1985) trustworthiness is critical in establishing the quality of a qualitative research study. Trustworthiness is the degree of confidence in data, interpretation, and methods used to ensure the quality of a study (Polit & Beck, 2018). According to Leedy and Ormrod (2005), trustworthiness is the extent to which other people find the results convincing and worthy of acceptance. Credibility, transferability, dependability and conformability are the four strategies that increase the trustworthiness of qualitative research

(Lincoln & Guba, 1985). Using the guidelines provided by Shenton (2004), the following study met these strategies in the following ways:

- The credibility of this study was enhanced by providing rich text descriptions of the phenomena and a proper explanation of the qualitative process followed.
- Transferability was achieved by providing rich text descriptions.
- Dependability and confirmability were achieved by providing details of the qualitative research process.

4.5.3. The Quantitative Phase

During the quantitative phase of this study, the researcher wanted to validate the revised conceptual model with a large population of software professionals and determine the critical acceptance and success factors. Therefore, the survey method, which used a questionnaire, was deemed to be the most appropriate.

Questionnaire Development and Organisation

The current literature on acceptance and success factors underpinned the development of the questionnaire. Studies such as Chow and Cao (2008) and Chiyangwa (2017) were used to develop and adapt the questionnaire for the DevOps context. The main data collection instrument used for the quantitative phase was strongly aligned with the UTAUT2 model. UTAUT2 is a well-established technology acceptance model with a validated questionnaire that was adapted for the purpose of the current study. The original UTAUT2 questionnaire was adapted based on the knowledge obtained from the study's literature review (e.g. content from Sharma and Coyne (2015) was also incorporated into the questionnaire development).
The questionnaire in the quantitative phase of the study consisted of closed-ended questions and was divided into three sections (Appendix B). These sections are:

- Section A – Captured the biographical information of the respondents, including work experience in DevOps, role in DevOps, and type of organisation they work for.
- Section B: Contained Likert scale questions about acceptance factors
- Section C: Included Likert scale questions on success factors.

Quantitative Sampling Strategy

For quantitative studies, sampling approaches can be divided into probability and nonprobability sampling. Probability sampling is used when the researcher generalises the

results to an entire population. However, the difficulty in carrying out probability sampling is establishing a sampling frame that allows respondents to be randomly selected for the study. A more time and cost-effective sampling procedure is non-probability sampling. Since the researcher wanted to target professionals with experience working with DevOps, purposive sampling was used as the sampling technique.

The Sample Size

The number of practicising DevOps professionals in South Africa was difficult to ascertain because of the wide variety of contexts in which DevOps may be employed. Devops is also a relatively new strategy that underpins software development and may organisations may be resorting to a DevOps approach in an exploratory or phased-in manner without making a definite claim to use the methodology. Hence, the identification of a sampling frame for the study was impossible to acquire. Hence, the population size for the study unknown, thereby making the identification of a discrete sample for the study difficult to achieve. While the inability to define a specific sample for a quantitative study is usually viewed as a serious impediment for the study, the current study has to be viewed as an exploratory one where the identification of a well-defined population and sample are not critical to the objectives of the study. The study did however follow a scientific method of inquiry and made liberal use of descriptive and inferential statisitics to validate the study's discussion. The technique of Structural Equation Modelling (SEM) analysis was used to validate the revised conceptual model used to guide the quantitative phase of the study. The discussion pertaining to sample size has been largely guided by the number of respondents suggested by researchers for SEM. A common view adopted is that of Kline (2005), who suggests at least 200 respondents are required in a quantitative study to enable SEM. This number became the minimum target for this study.

Quantitative Data Collection

Questionnaires were initially physically distributed and collected from respondents to increase the response rate for this phase of the study. However, there was a COVID-19 outbreak during data collection, and the data collection strategy was changed to online. Only 46 valid questionnaires were collected using the face-to-face strategy. In order to facilitate the online data collection, a Google Survey Form was used to create the questionnaire. The researcher leveraged the position of being a member of an academic institution to contact Alumni of the

institution who were registered for the postgraduate module in Software Engineering offered by the Information Technology department. Furthermore, many respondents were recruited via LinkedIn. DevOps professionals working in South Africa were sent a LinkedIn request to participate. If they agreed, an email was sent to explain the purpose of the project and a link to the questionnaire.

Quantitative Data Analysis

Data was collected via a closed-ended questionnaire and was coded and entered into the quantitative data analysis software package named SPSS version 29. Hereafter, biographical data was analysed using frequency calculations that were illustrated with histograms. The acceptance and success factors items were also subjected to frequency analysis. The strategy was used to provide the researcher with a holistic view of the data. Aggregated values such as means and medians were used to determine whether each factor exhibited a level of statistical significance ($p<0.05$) to determine if the phenomenon under inquiry showed a significant level of agreement or disagreement. This was ascertained by making extensive use of the Wilcoxon signed-rank test. The study's hypotheses were also tested via the technique of structural equation modelling (SEM), which was implemented via the statistical package SPSS in conjunction with the add-on AMOS version 29. Before measuring this revised model's structural path, the data was first subjected to exploratory factor analysis (EFA) and confirmatory factor analysis (CFA) to determine if the study's data has an optimal alignment with the study's conceptual model. After the valid measurement model had satisfied the construct and determinant validity requirement, it was converted to a structural path model and analysed.

Quantitative Data Validity and Reliability

Content validity determines if the items measure the construct accurately (Creswell, 2012). This was first achieved using studies that previously measured most of the constructs. After that, the instrument was subjected to a pilot study, which was given to three DevOps experts to assess whether the questions accurately measured the constructs. The questionnaire was corrected based on the feedback from the pilot study. The feedback was mainly on the phrasing of the items to align better with DevOps. Other reliability and validity tests were performed using CFA and discussed in Chapter 7.

4.6. ETHICAL CONSIDERATIONS

According to Creswell (2013), researchers must respect the ethical considerations of participants. This study obtained ethical approval from the University of KwaZulu-Natal Ethics Committee (Appendix C for approval letter). Furthermore, during the data collection phases, the participants' consent was obtained by signing an informed consent letter stating the study's main objective and that they were willing to participate. During the data analysis phase, no names of the participants were mentioned in the write-up to maintain the privacy and confidentiality of the participants. A number (e.g., Participant 1, Participant 2) was used as an alias to preserve the privacy and confidentiality of interview participants.

4.7 CONCLUSION OF THE RESEARCH DESIGN

This chapter highlighted the methodological design and methods used in this study. Since this study design consisted of both qualitative and quantitative phases, a sequential research method was selected. Qualitative data was analysed via Nvivo using content analysis, and quantitative data was analysed via IBM SPSS and IBM AMOS Version 29. The current chapter has elucidated the study's research design and methodologies. The following three chapters will be used to present the study's data and the analysis. An inspection of the illustration in Figure 4.2 will provide the detail regarding the content of each chapter. Figure 4.2 also provides an overview of the sequence that will be followed in the data analysis and presentation.

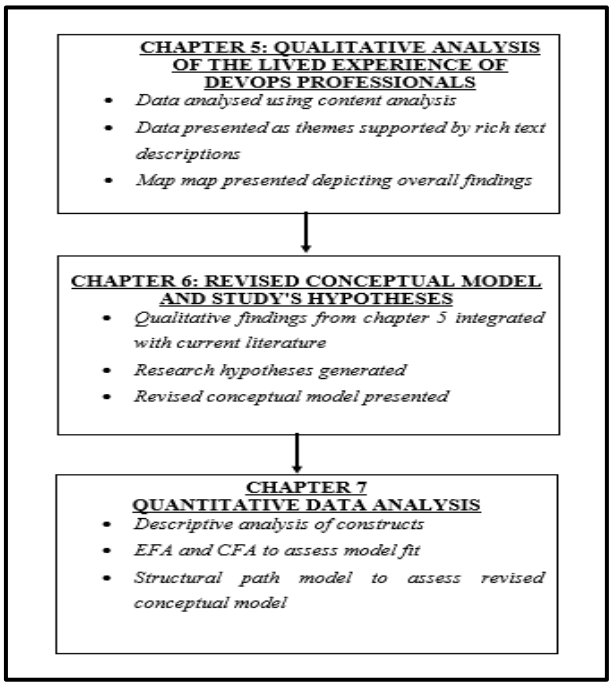

Figure 4.2: Outline of data analysis chapters

CHAPTER 5: QUALITATIVE ANALYSIS OF THE LIVED EXPERIENCE OF DEVOPS PROFESSIONALS

5.1. INTRODUCTION

The previous chapter elaborated on this study's research design and methods. The sequential research approach highlighted in Chapter 4 includes a component that entails a qualitative examination of the experience of systems development by DevOps professionals. This component of the study provided the researcher with an opportunity to obtain a meaningful insight into DevOps implementation from experts who have established familiarity with DevOps in a professional working environment. The research questions that guided this aspect of the study are listed below:

1. What are South African software practitioners' perspectives on the DevOps phenomenon?
2. What acceptance factors influence software professionals' intention to use DevOps?
3. What success factors influence DevOps project success?

Semi-structured interviews were used to collect data, and the existing body of research on acceptance and success criteria served as a lens through which the interview questions were framed. Phenomenological qualitative research was used for this study, and content analysis was utilised to interpret the data gathered. In order to document the lived experience of participants with a phenomenon, this method extensively uses textual descriptions, which is an essential aspect of phenomenological data analysis (Alhazmi & Kaufmann, 2022). Therefore, to explain the lived experience of software professionals concerning DevOps, this chapter contains a detailed written description of the responses made by the participants. This chapter presents the findings of the participants' definition of DevOps, the participants' opinion of the strengths and challenges of DevOps, and the acceptance and success factors of DevOps.

5.2. RESULTS

5.2.1. Demographic Details

All of the participants were from the banking sector. Of the nine participants, eight were male and one was female. All participants have experience working in software development and DevOps environments. Hence, they met the profile needed to provide a lived experience of using DevOps in a professional environment. The strategy of focusing the study on the banking

sector is driven by the knowledge that many technological innovations within South Africa arise from the banking sector. According to Romagny (2024), South Africa's digital banking is world-class and was ranked 2nd on Oliver Wyman's digital banking index, outperforming countries such as the UK, Spain, and Germany. The index assesses the digital capacities of over 100 banks in 11 countries using over 300 measures. Hence, these participants from the banking sector were ideally positioned to provide in-depth knowledge of the DevOps phenomenon. As indicated in Table 5.1, the roles of the participants included software and systems testing, IT security, developer, DevOps engineer, DevOps transformation lead, and DevOps operations. Their years of experience in DevOps ranged from 2 to 5 years.

Table 5.1: Details of Participants

Participant pseudonyms	Role	Gender	Years of Experience in software development	Years of Experience in DevOps
Participant 1	Testing	Male	8	3
Participant 2	Operations	Male	5	3
Participant 3	DevOps Engineer	Male	25	4
Participant 4	Developer	Male	25	3
Participant 5	Security	Male	10	4
Participant 6	Developer	Male	5	3
Participant 7	Data Engineering	Male	12	3
Participant 8	DevOps Transformation Lead	Male	15	4
Participant 9	Transformation Lead in IT	Female	28	3

5.2.2 Content Analysis

Chapter 3 laid the initial foundation for this chapter with the development of a preliminary conceptual model that highlights the main factors that affect the acceptance and success of DevOps adoption. These concept driven factors were based on prior themes established in the literature as indicated by the UTAUT-2 model for acceptance factors and by Chiyangwa (2017) and Chow and Cao (2008) for success factors. The majority of the questions for the semi-

structured interview schedule were framed around these factors. After phenomenological data was collected, it was transcribed and verified before being subjected to content analysis. These findings are discussed below, initially with a table highlighting the themes, followed by a discussion and rich text description.

Definition of DevOps

Participants were asked to define DevOps based on their experience with DevOps. Table 5.2 highlights the main themes derived from the participant's definition and understanding of DevOps during the interview process.

Table 5.2: Themes pertaining to DevOps Definition

Category	Themes
Definition of DevOps	Culture of collaboration
	Utilisation of tools
	Automation
	Processes and practices
	Ownership of code
	People

In order to develop a context for the presentation and discussion of the interview responses, participants were required to provide a narrative or an intuitive explanation of their understanding of DevOps. This component of the qualitative data analysis provided the researcher with an opportunity to gain insight and acquire knowledge of metaphorical references as well as technical jargon that was commonly used by DevOps professionals. As an example, Participant 9 alluded to DevOps as *breaking the wall* between development and operation teams so that everyone is responsible for the success or failure of implemented features. In terms of an actual definition of DevOps, the response was more formal, as indicated in the verbatim extract that follows:

> *…as software engineering or application teams merging with operational teams to be collaborative and more fully functional and own the entire life cycle or a value stream chain for a product or service within the bank* ***(Participant 9)***.

The lack of consensus and a concise understanding of the strategy of DevOps becomes apparent when the response from Participant 1 is analysed. According to Participant 1, he had to look up a definition on Google to comprehend what it meant. Participant 1 was employed in the capacity of a software tester. However, he was thrust into a DevOps environment as a consequence of his organisation's imperative to adopt a DevOps approach for all software development projects. His understanding of DevOps evolved from being a novice DevOps practitioner to one where he has acquired sufficient experience to be knowledgeable of the strategy. From an overview perspective, Participant 1 was of the opinion that DevOps could be defined as a combination of development, operations and quality assurance, the latter of which alludes to the capacity that he provides for his organisation. The response from Participant 1 suggests that while DevOps has a strong focus on software development and operations, the DevOps process-cycle has to include an intensive software testing phase as well as the integration of quality controls to ensure the development of quality systems for the production environment. The response from Participant 1 is provided in the verbatim extract that follows:

> *You know the first introduction to the word DevOps you have to Google it to be honest. To Google it to understand what it actually means. Development Operations and especially if I look at my designation coming from a quality assurance which is the old testing background. You wonder where do you fit in with this whole thing. And if you unpack the word development, then you unpack the word development seeing that quality plays a huge role in the development process as well as the operational process. So that is where my introduction to DevOps came about. Having to understand that where do I as a traditional tester come into play and we play a very significant role in both the development and the operations and putting the DevOps together. Even from a development piece making sure that the code is of quality and from an operations perspective as well that if your code is of good quality obviously traditional whatever goes into production is of good quality. So you are able to give both input from a development perspective as well as input in an operations perspective* ***(Participant 1)****.*
> .

When describing DevOps, Participants 2, 4 and 7 brought up the concept of automation, indicating that DevOps also encourages the automation of the development stack to achieve more reliability in the production environment.

> *My understanding of what it is, is typically you've got developer processes and methodology to develop your software and build an application. Then Ops would be providing the infrastructure and the environment in which*

> *to run that application, and they're basically, two, totally separate disciplines but for you to actually take your application and implement it on infrastructure... that becomes then the DevOps paradigm. But the actual process of that would be to automate the different facets of that as much as possible* ***(Participant 2)***.

> *DevOps to me is. It's a method on which you can systematically and automatically just run your change or run your code to enhance or to ensure production stability* ***(Participant 4)***.

> *DevOps is the process of handling delivery (BAU or enhancement) and operation by the same team through set of tools available in the market through feature team model. It also supports, automation of software package creation, deployment into respective environment, unit testing and then promoting the code into respective environments (E.g. Application testing, integration testing, etc.)* ***(Participant 7)***.

Participant 5 asserted that the definition of DevOps incorporates people and technology into the DevOps strategy. This guarantees access to the relevant tools and individuals with the required skill set.

> *From a people perspective, I think it's about getting the right people who would normally do the development to actually look after the systems as well. From a technology perspective, it's the toolsets that enable the people to do their jobs so, the development and the supporting of production* ***(Participant 5)***.
> .

In addition, Participant 6 defined DevOps by stating that it is comprised of the proper procedures and practices that assure effective governance across the development and deployment lifecycle.

> *Above the traditional definition obviously rather I'd rather take how I'm implementing it or the team is implementing it itself. It is more of a practice that allows the smooth flowing of a feature from its inception until it goes straight into production. So, from not only just the tools but the processes we follow within the team considering as well all the CAB processes you have to take into account* ***(Participant 6)***.

Finally, Participants 3, 5 and 8 indicated that DevOps meant that the team took complete responsibility for the code, from creating to deploying and maintaining it.

For me it's simple. You build it, you run it, you own it so, it's exactly that. I would go one step further on top of DevOps and I would say, it's full stack engineering ***(Participant 3)***.

The same people that built it are supporting it. They understand it from start to end, rather than just throwing it over the fence and another application team needs to look after it so, it simplifies the process... you build it, you own it. If you look at what we've done in the card space, you will have an infrastructure guy, you'll have an application guy, a security guy, database guy, a storage guy, etc... all part of the same team and they are part of that whole process. So, they build it and support it ***(Participant 5)***.

They wrote their code, they test their code, they deploy their own code and then they support their own code. Okay, there's no handovers ***(Participant 8)***.

The verbatim textual extracts obtained from the interview responses provided a valuable basis from which a "working definition" of the strategy of DevOps could be established. However, the strategy of engaging these professionals in an open conversation on the topic of DevOps also provided an opportunity to identify prominent words and phrases that have been used as a set of classifications or themes to add structure to the presentation and discussion of the qualitative data. The set of themes pertaining to DevOps has been identified as automation, the culture of collaboration, code ownership, processes and practices, and the use of tools and people. The frequency of references to these themes/classifications is illustrated in the histogram shown in Figure 5.1.

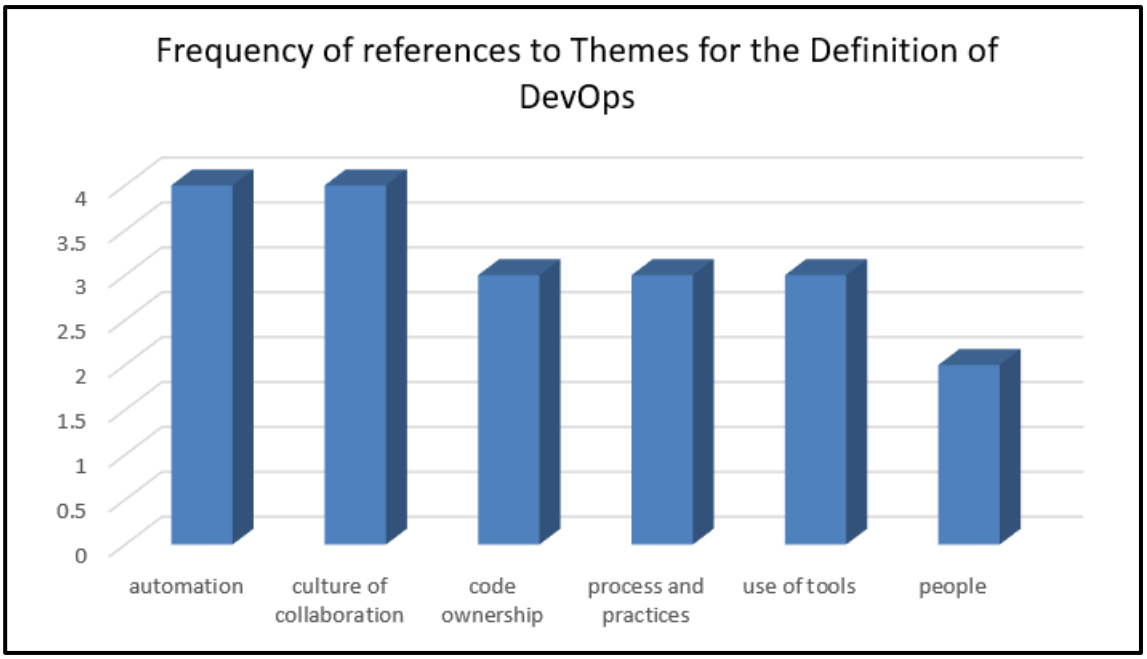

Figure 5.1: References attached to the Identified Themes

According to Saldana (2009), once themes have been identified, the next step is to use these themes as a basis from which to develop a holistic conceptual understanding of the qualitative data. This can be achieved by leveraging the themes that have been identified to craft a textural description of the phenomenon being analysed. In the context of the current study, themes that emanated from the discussion pertaining to the definition DevOps have been crafted into a cogent statement that reads as: DevOps entails a merger between *development and operations, that results in a culture of collaboration that can be improved by utilising tools and automation within well-defined processes and practices*. This helps to maintain the quality of the software development process and establish code ownership.

Strengths of DevOps

The major themes in Table 5.3 summarise participants' experience regarding the strengths of DevOps.

Table 5.3: Themes pertaining to strengths of DevOps

Category	Themes
Strengths of DevOps	Faster delivery
	Greater consistency
	Better knowledge sharing
	Ownership and accountability
	Better understanding
	Improved product quality
	Early problem identification and resolution
	Greater project visibility

Faster delivery

On the topic of the strengths/advantages of using a DevOps strategy, participants 1, 3, and 6 believed that DevOps makes it possible to deploy features more quickly, with a higher level of quality, and with a better alignment with the consumers' requirements. The preceding claim is corroborated by the verbatim textual extracts referenced by each interview participant.

My view is obviously... if you look at the traditional way of doing things. Which is waterfall it took too long and my view of DevOps is how we are able to resolve the mess that happens in IT organisations in delivering widgets if I may use the word, to the customer. It is important and the only way you can do it is if you do it much faster, better quality and something that is for the customer. You can't just give a customer a widget that the customer didn't want and the mere fact with the DevOps, the whole DevOps journey is you are more able to be more close to what the customer is wanting. And if I'm saying the customer, that is the business because the business is the one that gets the information from the customer. So if you are able to playback to the business which is the eyes and ears for the customers much quicker and deliver much faster it could show the customers that this is essentially what the DevOps is about ***(Participant 1)***.

I think we've got the stats to back it up. It's definitely made a huge difference. It's really dramatic what we've seen in terms of the average, and please, I don't have the exact days at hand but it was something like, on average, a couple of years ago it took 112-days, for example to land a typical project. Where I think last we checked it was 30-odd days now so, that in itself, speaks volumes. I believe it's been successful. I also like the saying where I don't believe we are where we want to be. Nowhere close, but we're also not where we used to be. Yes, we make mistakes. We've got a lot to learn still but the trajectory is heading in the right direction ***(Participant 3)***.

We still at not infant but maybe toddler stages. I know the teams with ... my overall view is the teams do use it. not all use it as is intended. Others use it not only to further their Ops portion. They'd focus mostly on the ops others would still create those silos, but overall in terms of delivery for those that are using it correctly it has shown to have great improvements from the entire SLDC… that improves their deployment times and testing's and all that ***(Participant 6)***.

Participants 5 and 7 clarified that the operations teams play a crucial part in achieving faster delivery in DevOps by ensuring that the full development stack is supported, beginning with development and continuing through maintenance.

The first few occasions were difficult. The first build that we did took us more than a year but it was infrastructure application, database... everything. Compared to now, where the guys are taking months to do it. Where they can build an entire stack within, I think the best time was about 34-minutes. Putting down an entire application and this is a complex card application so, it's being good ***(Participant 5)***.

...have quite improved the delivery. We basically for a ... we build the railway line and the team is that train that is just moving towards a certain destination. We are the ones that provide that railway line from building

that environment, deploying the basic dependencies and liabilities onto those environments to the final application in the testing environment until it goes to prod and also rather nor or, and also maintaining that. So it has greatly improved the teams delivery ***(Participant 7)****.*

Greater consistency in the development environment

An explanation was given by Participant 2 regarding how DevOps made it possible to have a more consistent development environment that included all of the necessary dependencies and memory needs. This ensured that the feature would not fail when it was deployed.

So, for me, the top thing is to be able to consistently create your environment because every time that you deploy or make enhancements to your application. In the past, there'd be little things changed on the server and you don't always know that and then your production gets depleted. So, your infrastructure is not necessarily consistent and it becomes almost like a old cow. It's like that is the server and you can't log into this thing because you can break this and that. Where now, if you have a pre-defined server or servers because it'll be the application on the database with however many different parts you have to be able to create each one of them consistently, is to me, one of the big benefits ***(Participant 2)****.*

Better knowledge sharing within the organisation

People are required to document and share their knowledge as part of the DevOps methodology, which encourages improved knowledge exchange throughout the firm. This is significant, particularly in IT, since individuals who possess specialist knowledge are in high demand and can quickly find another work in a new organisation or division, taking their experiences and knowledge with them. This knowledge management aspect of DevOps is seen as a benefit according to participants 2 and 5.

...your knowledge share as well, is you used to have these few individuals that's got their expert stuff. They've got their notes and everything taken and these guys can do magic because they know what they're doing. Nobody else knows what they do but they do so, you've got this key-man dependencies but with the DevOps you're forced to actually, put that knowledge in a source that everybody can see ***(Participant 2)****.*
So, the communication increase... that's been phenomenal. Then the skills that come from each area. Your production troubleshooting skills, which

is normally your production resources from operations that's they do firefighting the whole time so, you get a very specific set of skills and then your developer insights and then combining those two and sharing that... that's been the strongest benefit of DevOps ***(Participant 5)***.

Better sense of ownership and accountability

Before DevOps, when developers complete coding a feature, it is thrown over the wall to the operation teams for deployment. This enables them to move to a new project, relinquishing themselves of further involvement in that feature. When there is a problem, it becomes a game of assigning blame, ultimately resulting in a delay in the deployment of the functionality. On the other hand, in DevOps, every team member takes ownership of the product, and the team is responsible for ensuring that any problems are resolved, as asserted by Participant 3.

The whole dynamic changes with DevOps. If you've got an engineering team or a software engineering team that's writing to code and if they know that they are going to be the ones that are going to be phoned at 02h00 if something goes wrong then it kind of puts a different spin on things. Those days of writing software, throwing it over the fence for somebody else to support it is gone. So, for me, that's the most important factor that stands out, in terms of my experience with working and leading DevOps teams is really that sense of ownership and purpose. We talk about autonomy mastery and purpose and for me, you can't have that without a DevOps team ***(Participant 3)***.

Better understanding between different stakeholders

Few interactions took place amongst stakeholders as a result of the fact that various departments within software development were operating in their own pockets of space. As a result, individuals did not know the roles, duties, and difficulties faced by other stakeholders. Participant 5 said that when many stakeholders got together as a team in DevOps, there was more collaboration, which promoted a better understanding among the various stakeholders.

I'd say a better understanding. You now have your infrastructure and your software developers in the same room. They can now understand each other better because you start talking the same language, there's not an us an them. It's collaboration but you get a better understanding of what effort needs to be done on this side, versus that and it's a much easier way of working. To me, it's faster ***(Participant 5)***.

You get a better understanding and faster as well because the guys are sitting with one another. It's not a question of a mail or make a call and

> *now I have to wait because Harold isn't there at his desk, or he's in a meeting. 'I'll only get back to you tonight or tomorrow.' So, it's definitely faster* ***(Participant 5)****.*

Improved product quality

If clients are to embrace a new system effectively, developers must ensure that the non-functional are also implemented. The success of a system is dependent on several non-functional requirements, including but not limited to security, capacity, compatibility, maintainability, scalability, and performance. According to Participant 5, non-functional criteria are considered considerably earlier in the development process under the DevOps methodology, which ultimately results in improved product quality.

> *By being in one team and having the roles, you think about the non-functional requirements a whole lot more. That's normally not a consideration for development but it plays a big part in your production environment so, that gets a lot more attention than it used to get* ***(Participant 5)****.*

Early problem identification and resolution

According to Participant 1, DevOps makes it possible to deploy code into the operational environment earlier, allowing the opportunity to identify issues earlier. Because of this, the team can correct faults earlier and gain invaluable knowledge from these failures much earlier.

> *We start something, we see, we've learnt, there is a problem we fix that problem and make sure we don't fall into the same problem in the next step. That is I think what we need to take out of this whole thing. Is that we must learn from our mistakes sooner rather than later and I think that is also the nice thing about DevOps, you need to fail fast not fail down the line. Able to strap our boots up together and say okay fine we know we stuffed up there. This is how we need to go going forward and I think it is a learning process. Waterfall couldn't allow you to learn from... you learn when the thing is already in production and that is way too late. Here you are able to learn in the beginning and able to remedy that whilst going ahead* ***(Participant 1)****.*

Greater visibility in the projects

The deployment of software typically takes place all at once towards the end of the project when using the waterfall methodology. In light of this, a more accurate depiction of the result

is typically observed in the latter phases of the cycle. Using DevOps with Agile, the first participant stated that team members could monitor the project's progress during the entire system development lifecycle.

> *Strength of the process definitely remains the visibility of it and the ease of shipping. Shipping... the ease of being able to now see. Not just one person having a project plan and doesn't share the project with the greater community. You are part and parcel of a particular requirement moving from inception to testing, to development, to unity, to production. That is the nicest thing you can actually see the moving target. You can see okay fine this is... traditionally you wouldn't be able to see what is the impact if something doesn't go in. Now it is much more clearer that you can see that this doesn't go in this is the impact. And the mere fact that you know there is something else waiting down the line that you can bring in and you can take that particular widget that you are shipping out and putting another widget that will serve the same purpose, but not the same outcome. That is what I like about the whole DevOps thing* ***(Participant 1)****.*

Challenges to DevOps Adoption

Participants challenges with the adoption of DevOps is summarised by the major themes in Table 5.4.

Table 5.4: Themes pertaining to challenges of DevOps adoption

Category	Themes
Challenges	Legacy code
	Too much focus on delivery
	Automated testing lag
	Not one size fits all process
	Learning, unlearning and relearning
	Cost

Legacy Code

According to participants 1 and 4, it is a challenge to implement DevOps in environments that have legacy code. Due to the monolithic nature of legacy code, it becomes difficult to separate this code into features to apply the DevOps strategy. DevOps is ideal for developing software

features using micro-services, which is challenging to achieve from legacy code. The COBOL programming language creates only structured code, preventing code from being packaged into different components.

> *If you look at our legacy systems. Legacy systems definitely. Look at how many organisations have 3270s, how many organisations still have batch systems. And I think the ones that are away from legacy are performing much better* ***(Participant 1)****.*
>
> .
>
> *What I find is the method on new age technology is a lot easier than what available for your Legacy technology, like COBOL. The reason why I say that is, like for instance when you develop in Java. You have your main class, and from your main class, you go into a multiple other children class. And when you're writing your software it's easier with DevOps to go and kick-off a certain class and you know exactly where your problem lies. COBOL is slightly different in a sense that that you don't have a main class that you go and kick off some of the other classes. You have a program and it's developed in a structural manner so, for us to do DevOps is slightly more difficult than what it would be for your new age technology, your object orientated programming* ***(Participant 4)****.*

> …using DevOps on your Legacy systems is a lot harder than your new-age technology. So, I think in your Legacy systems the chances of DevOps dying a slow and painful death. However, with the object oriented programming and your new age technology... I think DevOps plays a massive role ***(Participant 4)****.*

Too much focus on delivery

Even though one of the key advantages of DevOps is the ability to bring products to the market more quickly, Participant 6 believed that placing excessive emphasis on delivery could be an issue. Software developers tend to become so preoccupied with the delivery of their work that they fail to consider ways to enhance the process and eliminate any steps that are not necessary. Furthermore, the focus on delivery prevents the developer from thinking about the long-term maintainability of the features being developed.

> *There is lots because remember you want to divide into feature teams right to work into agile. You want to make sure that your entire DevOps process is actually working. You want to make sure that the development is actually fine and you want to deliver as quickly as possible to market. So because you're always on that delivery mode you tend not to take a lot of time focussing on okay let's stop, let's now rethink. Sure there are retrospectives but in my experience retrospectives have not been very*

helpful. So people will look at them for those first two days but after that everybody goes back into delivery mode. There is never anybody that says take this time, clean up your processes and see how much wastage we have right ***(Participant 6)***.

so delivery mode is actually killing the whole, I would say it is actually killing. Because I believe there should be a proper balance. Deliver and make sure you maintain. Right. If you can't maintain you will constantly be pulled into support tasks. This is going wrong, this is broken, this is happening, this, is happening and nobody wants that. Nobody wants to form ... nobody wants to be that guy who is sleeping and gets called in the middle of the night because something broke. If we take the time to actually must clean processes out it would be much better. So that is the challenge that comes along, always in delivery mode and not taking the time to clean up and fix whatever needs to be fixed before it spends its entire life time in production and then you find out only after a year that something is not right. Or maybe after two weeks that something is not right. Now you have to go back and redo. So that is one of the biggest challenge ***(Participant 6)***.

Automated testing lag

Participant 5 claimed that automated testing in DevOps is still a difficult task to do. Tests are still carried out manually or with the assistance of test scripts. There is still a lack of cost-effective consolidated testing environments. Additionally, establishing a framework for test automation is time-consuming and requires a great deal of technical expertise.

Part of the testing process? So, we haven't got an automation testing down to working. That's one of the stuffs that we're lacking but we are part of the testing process. We'll develop test cases or we'll recommend 'guys, you need to look at this or do that... this is what you need to look for and test for.' But we don't have our automated testing process down to a T... that we're still lagging behind ***(Participant 5)***.

easier access to our testing capability now, but it's not fully automated. So, the ability to run fully automated test scripts and get green ticks and that type of thing is still on the cards for us, and something like that would be phenomenal. We've never had it before. So, a lot of the testing is still manual. Yes, everyone has their own scripts that they run but not a consolidated environment for automated testing. The environments that exist can be very costly. So, that's still coming and is therefore a challenge when adopting DevOps ***(Participant 5)***.

Not one size fits all process

Participant 1 indicated that it is not a one-size-fits-all process but needs to be adapted to the organisation and type of projects. It is also essential that when DevOps is implemented, it is implemented on a small scale first. This allows the organisation to learn from mistakes before moving on to a large-scale deployment.

> *The biggest challenges...it's not a one fits all. It is not a one fits all kind of methodology or process. You actually need to see where it will fit in the best. Start off with a small project. Get a proper case study if I may use the word of what was the pros and the cons. You must pilot it first and see what is the benefits and you learn from those mistakes. Because that is the definition of DevOps. We start something, we see, we've learnt, there is a problem we fix that problem and make sure we don't fall into the same problem in the next step* ***(Participant 1)****.*

Learning, Unlearning and Relearning

According to Participant 3, software professionals persist in remaining confined to their sphere of expertise and are unwilling to broaden their horizons and acquire new knowledge, even if doing so could favour the project as a whole. In light of the fact that tools and technologies make it possible for developers to provision their infrastructure through the use of infrastructure as a service, this might be a challenge for DevOps, if developers are unwilling to learn. This can have a negative impact on the project.

> *A couple of years ago, two or three years ago, I had a developer in the COBOL space in a room like this and we were going through a demo of a new product tool set and the ability it gives you to actually publish your services, internal or external, how do you get people to subscribe to your API? How do you manage the throttling and the performance thereof, and so forth? She said something that I will never forget, she said, 'I don't care about any of that... I just want to write my code.' I couldn't believe it and I said, 'sorry guys, but there's no space for that.' You cannot tell me that you do not care about that. It's not somebody else's problem... it's our problem. So, yes, that example comes to mind* ***(Participant 5)****.*

Further investigation revealed that Participants 3 and 4 concluded that the difficulty of learning is not necessarily the issue at hand; instead, the issue is the difficulty of unlearning. There are a lot of older developers who are unwilling to unlearn what they have learned in the past by

choosing not to adapt to and improve their knowledge of contemporary tools and technologies. This is why recent graduates are better suited for DevOps since they are more eager to acquire new knowledge and remain current with technological advancements.

> *Well, learning is not necessarily the problem. Un-learning is the problem. Getting rid of the old-ways of thinking. Typically what I'm seeing more and more. Bringing in the grads, give me the grads because these guys they are a blank canvas. You can mould them. I'm not dissing anybody. We are literally standing on the shoulders of giants. They got us here so, we need those deep, intrinsic SMEs, those 30... 35-years in a specific field. COBOL or Dev... they know that system back to front and front to back. My problem is trying to get those guys to realise that we are part of a modernisation journey, for example, and in three-years your system will not exist. This application will run on a different tech stack. It will no longer be COBOL, and that's my problem. It's that willingness to be agile* ***(Participant 3)***

> *From a Legacy point of view I think that we need to change our mindset and the way that we develop within our Legacy systems, in order to go forward with it. I think that is one of the weaknesses in that your Legacy developers are all your older developers and their mindset seems to be a lot harder to change than your youngsters. So, your youngsters are quick on their feet. They're quick to change their mindset whereas your older generation, who are Legacy developers, they tend to stick to what they know* ***Participant 4).***

Cost

Participants 1 and 9 both mentioned that the expense of implementing DevOps can be a barrier to its widespread adoption. Investing in the appropriate tools and technologies can be a costly endeavour. One method to circumvent this challenge is gradually introducing these technologies across the business. This can be accomplished by choosing tools to deliver the most significant customer benefit sooner.

> *Cost is most definitely a factor as well. Cost is definitely a factor but if small start-ups can do a lot with less surely a lot of other companies should do the same as well. If you... how do you eat an elephant, one bite at a time. We need to start doing things bit by bit, able to get it to a customer much sooner in the way that they want it and not in drips and drabs.* ***(Participant 1)***

> *Part of DevOps can be implemented by having greater collaboration between different teams in the organisations, but to get the greatest benefits of devops is tools and automation and depending on the toolset you are targeting it can be an expensive game. You do get the open source*

tools but they may not be compatible with you're the technology platform in the organisation and maintenance of these tools can be an issue. ***(Participant 9)***

Acceptance Factors

Performance Expectancy

The degree to which an individual believes that DevOps is useful is evaluated through the use of the performance expectancy factor. Table 5.5 illustrates the themes emanating from this factor.

Table 5.5: Themes pertaining to the factor Performance Expectancy

Factor	Themes
Performance Expectancy	quicker completion of projects
	reduced project comebacks
	improved individual productivity
	improved team productivity

Concerning the usefulness of DevOps, the participants' perspectives are documented through the Performance Expectancy factor. Within DevOps, performance expectancy is defined as professionals' perception of using DevOps to improve one's job productivity and attain positive software development project outcomes.

According to participants 3, 5, 6, and 8, DevOps is useful because it facilitates more collaboration between teams working in diverse domains (such as development, security, testing, and operations), ultimately resulting in improved overall success in software development projects. The following are the thoughts of software professionals that suggest the speedier resolution of problems due to the cooperation culture incorporated throughout DevOps. This results in increased productivity and project success.

It definitely had a positive impact on the outcome of the projects. With increased collaboration among individuals with different skills, problems are resolved faster leading to better project outcomes. It improved individual and team productivity ***(Participant 3)***

Previously each of the domains were on their own island. They only come together to solve problems that may arise. Now with teams coming together earlier, we identify risk of project failure earlier together and can

put plans in place to mitigate them. Projects are development and deployed faster and there are less comebacks with our projects like before ***(Participant 5)***

It is definitely useful. Before I use to develop software and give it to the operations division only to have come backs like incorrect dependencies files or missing files, etc. With DevOps and close alignment with operations, we develop more on production-like servers which means that the software has a less chance of failing in the operational environment ***(Participant 6)***

Sometime people view collaboration as talking more and doing less. But in DevOps there is a diverse array of skills and knowledge and all are needed to make the project a success. So the collaboration in DevOps paves the way for more individual and team productivity ***(Participant 8)***

Implementing a wide range of tools and technologies inside the DevOps environment proved to be quite beneficial in enhancing the efficiency of the software development process and achieving the desired outcomes for the project, as stated by Participants 2 and 4.

DevOps has a positive impact in our organisation. I think the main reason for this is the tools used to create, deploy and monitor software. I would say that these tools is impacting tremendously on improving the quality of our projects and the speed at which things get done ***(Participant 2)***

Often said that people are more important than tools. From my experience, both are equally important. If you have the right people combined with using tools, it greatly enhances the continuous integration and continuous deployment process. This is why DevOps is successful in pushing code to production quicker ***(Participant 4)***

Both participants 1 and 6 made reference to the fact that automation is beneficial to the DevOps process. By utilising automation and technologies, the team was able to boost their productivity, which ultimately led to the delivery of features more quickly and with fewer errors.

Me as an individual has not seen the increased usefulness as yet in DevOps, since in testing we are still facing challenges with automating our testing process. But overall as a team, especially guys that are able to automate their processes, I would definitely said yes that DevOps increased the productivity of the team ***(Participant 1)***

In fact it is useful at various levels. Being a developer, at the individual level using tools and automation increased my productivity. Using

virtualisation and containerisation tools, I am able to create servers quickly, no need to wait for an operation guy to physically set up a server. At the team level, greater collaboration and effort from different domains increases the chances of project success and this impact the organisation in a positive way because our client, a particular business unit gets features not only more quickly but with less bugs and downtime. ***(Participant 6)***

According to the respondents, performance expectancy advantages are associated with using DevOps. The common themes regarding performance expectancy were quicker completion of projects, reduced project comebacks, improved individual productivity, and improved team productivity. The productivity of DevOps was accomplished through collaboration and the utilisation of technologies and automation.

Effort expectancy

The degree to which an individual believes that DevOps is easy to use is evaluated through the use of the performance expectancy factor. Table 5.6 illustrates the themes for this factor.

Table 5.6: Themes pertaining to the factor Effort Expectancy

Factor	Themes
Effort Expectancy	Enjoy technologies
	Compatible work environment
	Age
	Personality

Effort expectation refers to the ease with which a particular technology can be utilised. In DevOps, effort expectancy refers to the ease with which DevOps professionals became accustomed to working in an environment specifically designed for DevOps.

Participant 3 mentioned that his enthusiasm for acquiring new technological skills makes it easy for him to utilise DevOps.

I am a technical person. I love looking for new technologies to make my work more efficient and effective. For me, adjusting to the DevOps way of working was easy for me ***(Participant 3)***

.

Participants 4 and 5 reported that it was simple to implement DevOps because it was strongly connected with the agile mindset.

> *DevOps is one step forward from agile. Agile enables more continuous development of software and joined with the DevOps approach, these features are implemented more rapidly into the operational environment. So DevOps is more aligned to the agile approach that we currently use in this organisation since the practices and objectives are similar in both strategies. By already working in an agile way, it is easier to implement and adopt the DevOps strategy as opposed to working in the traditional waterfall environment* ***(Participant 4)***

> *We use the agile approach, Scrum in this organisation. DevOps practices are well suited to Scrum since agile also promote the use technologies, it just in a different part of the lifecycle. Daily Scrums meetings are similar in the DevOps environment except that the teams now have more expertise from other domains such as security and operations* ***(Participant 5)***

Participant 8, mentioned that the introduction of new tools and technologies into the development environment could make DevOps more challenging to employ at first. Research, alignment and learning of tools are required to complement the tools already in place within the organisation.

> *Depends at which stage of DevOps implementation you referring to. In the beginning as a developer you need to get used to the tools and technologies. You also need to research the tools and find the ones that aligned to your organisation. When you become accustomed to these, working in the DevOps environment becomes easy* ***(Participant 8)***

Participant 1 asserted that automated testing can be complex due to the creation of automated processes across the software development lifecycle.

> *Working in testing is a bit of a challenge since it is not a quick task to automate the testing process at various points in the development cycle. Therefore for us testers using the DevOps approach is not as easy and we still need to mature within this environment* ***(Participant 1)***

Younger software professionals who can adjust to different circumstances find DevOps an easy process. To a greater extent, younger people are more receptive to continual learning and innovation. On the other hand, the older generation is unwilling to change and adapt to new circumstances because they are still trapped in old tools and processes. Therefore, DevOps is

a challenge for these older professionals. The following excerpt from Participant 6, shown below, illustrates this.

> *There are a lot of older folks that are project managers within system development divisions. These project managers was tasks to make sure the project was delivered on time and the team get the necessary support. They are not involved in the technical development of projects. In the DevOps world, these project managers needed to be incorporated into teams and take a more technical role. This was difficult for them to move from a non-technical role to a technical role especially when they had not engage with technologies for some time. Older guys had problems adapting and therefore difficult for them* ***(Participant 6)***

In addition, participants 4 and 8 indicated that personality is a significant factor in determining how easy it is to use DevOps. Introvert professionals in DevOps may find the field challenging because of the frequent collaboration with others.

> *It also depends on who is willing to make the change and the character of an individual. Devops certainly make you collaborate more which means communicating with others more. I saw guys in our project team that found the DevOps philosophy difficult because they were introvert developers and kind of considered themselves those cowboy coders who like to code the whole day with interacting with anyone* ***(Participant 4)***

> *It wasn't easy in the beginning because people like titles and if I have a title I speak the loudest, forget the fact that we have the same... we at the same, same requirement. Have the same goal. It shouldn't be who speaks the loudest. If we have the same goal we should treat everybody on the same level. So that everybody understands what the success would be at the end of the day. Not for one person but for everybody* ***(Participant 8)***

Participant 2 and Participant 6 believed that the implementation of DevOps is made simpler by the presence of technologies and automation.

> *It is easy since we have the right technologies and tools to implement the DevOps strategy. We have tools to support the entire pipeline.* ***(Participant 2)***

> *With virtualisation and containerisation it was easy for me to set up servers and the automatic pushing of code make implementation quicker.* ***(Participant 6)***

Participants agreed that utilising DevOps needed little effort, particularly for persons who enjoyed working with technologies and were already working in an environment comparable to Agile. On the other hand, DevOps can be challenging during the initial adoption and when age and personality are considered. Tools and automation were found to make DevOps easier to use.

Facilitating conditions

The degree to which an individual believes that an organisation provides the necessary support to facilitate the usage of a system is evaluated through the use of the facilitating conditions factor. Table 5.7 illustrates the themes for this factor.

Table 5.7: Themes pertaining to the factor Facilitating Conditions

Factor	Themes
Facilitating conditions	Organisational investment
	Training
	Open source tools

Participant 2 mentioned that their firms had made significant investments in resources to guarantee that DevOps would be effective in their places of business. To do this, the appropriate tools were made available, and further promotion and support were provided for these tools to facilitate automation within the software development process.

> *My organisation has invested a lot of money into the DevOps approach to ensure that it is successful. Management matched the vision of DevOps with increasing the resources necessary for it to succeed. To gain the maximum benefit of DevOps, you need to promote the use of tools and automation, which our organisation invested in and promote. Having these conditions are important for the DevOps strategy to be accepted and be successful* ***(Participant 2)***

Participant 1 noted that in addition to the expenditures made on technologies and infrastructure, the organisation also made sure that appropriate training was offered to the development team in order for them to be able to benefit from these technologies immediately.

> *For DevOps to be successful, it takes more than collaboration. Although collaboration is the start of the journey for DevOps to be successful, technologies are a must, and our organisation invested in these*

technologies so DevOps teams have the necessary infrastructure for project success. Besides spending on technologies and infrastructure, it also makes training available so DevOps teams are knowledgeable on the processes and tools ***(Participant 2)***

.

Participant 4 stated that open-source tools, which are less expensive than proprietary tools, have the potential to offer the facilitating conditions that are necessary for DevOps. It was because of this that the firm was able to advance its DevOps strategy.

The necessary proprietary tools for DevOps can be expensive. My organisation promotes the use of Open-source tools that can still achieve the same outcome. Tools like Chef and Jenkins are open source tools used in our teams and are supported by our organisation. In this way we have the necessary resources to practice the technical aspects of the DevOps strategy ***(Participant 4)***

Participant 6 stated that his organisation has adopted SaFe, a framework that includes DevOps as an essential component. Employees who receive training in the framework also gain familiarity with the DevOps methodology. The organisation, therefore, supports the DevOps strategy in this manner.

The bank has implemented a new way of working with the implementation of SaFe. Within this framework, DevOps is implemented. Since employees of the organisation are trained in SaFe, they become knowledgeable of DevOps and this is one of the ways knowledge of DevOps spreads within the company ***(Participant 6).***

As a result of the many tools available to support various aspects of the process, the environment of DevOps will continuously evolve and develop further. Participant 8 referenced the DevOps engineers, who is well-versed in the entire process and researched new technologies to enhance their process and then shared them with the team. As a result, the DevOps teams is supported by newly developed tools and technology.

The DevOps engineers in our teams really understands the entire pipeline and they constantly investigate new tools that will automate our pipeline and make it more efficient. Any new tools they discover that might help, they passes it on to the relevant team member that will benefit from it. In this way we gain more information of what tools are available and how it can make a difference to our individual and team performance ***(Participant 8)***.

Participants stated that they had the facilitating conditions to use DevOps. These conditions included organisational investment, training, open-source tools, and a skilled DevOps engineer.

Social influence

SI refers to the process by which individuals modify their behaviour, update their views, or adjust their opinions as a direct result of their interactions with others in their social environment. Table 5.8 illustrates the themes for this factor.

Table 5.8: Themes pertaining to the factor Social Influence

Factor	Themes
Social Influence	Peers
	Competitors
	Social media
	Conferences

Participants 2 and 3 stated that the choice to transition to DevOps in their organisation was driven by top management and that they followed the methodology that was given to them.

> *For me social influence did not impact my decision as this was dictated higher up in the organisation. Management said we are moving towards DevOps and just had to adapt, learn and implement the practices.* ***(Participant 2)***

> *My organisation implemented it and we just adopted the approach. We were told that the company is moving to Safe and DevOps and this will be the new ways of working within the organisation. We were all given training on this new ways of working* ***(Participant 3)***

Participant 1 stated that they became familiar with DevOps through their conference participation, ultimately leading them to apply it in their respective organisations.

> *I would say attending the DevOps conferences influenced the adopting of DevOps. When you attend conferences and you hear your peers talk about their productivity gains because of DevOps, you want to try it out in your organisation. You do not want to be behind the curve ball and want to innovate and stay current, especially if your competitors are. It becomes a no-brainer but to try it* ***(Participant 1)****.*

Participants 4, 6, and 8 reported that their interest in DevOps was inspired by acquaintances working in other organisations, competitors, and social media platforms like LinkedIn.

> *There is always a degree of influence from others. You have friends in other organisations and they speak about how efficient their software processes are due to DevOps and you do get influenced by their reviews.* ***(Participant 4)***

> *You go into LinkedIn and see the jobs available for DevOps engineers and get curious and start reading about this process and technologies.* ***(Participant 6)***

> *Let face it, if you are IT professionals, especially in software development, tools are changing rapidly and you must be able to learn and innovate to keep up with these changes. Interaction with people outside your organisation certainly helps to keep you on your toes. Also, I work in the banking sector. It is highly competitive with banks competing against each other to be the most innovative and come up with new products to attract and maintain customers, especially on the online platform. So, you do get influenced by what your competitors are doing* ***(Participant 8)***

Some participants indicated that no one influenced them to use DevOps because the process was implemented by management. Most participants, on the other hand, stated that they were influenced by their peers, competitors, social media, and conference attendance.

Habit

Habit is the degree to which individuals tend to carry out behaviours automatically. Table 5.9 illustrates the themes for this factor.

Table 5.9: Themes pertaining to the factor Habit

Factor	Themes
Habit	Collaborative culture
	Process improvement
	Benefits of DevOps

Participant 5 stated that he first had a negative attitude towards DevOps and believed that it was a waste of time until he witnessed its benefits. Following that, he routinely utilise DevOps.

> *Initially DevOps was just an inconvenience in my life. Going to meetings, collaboration with other teams, etc for me was a waste of time. I just wanted to get my work done. However, when I saw the benefits that was achieved in terms of quicker development, more quality software and less problems, DevOps became part of my daily routine* ***(Participant 5)***

Participant 3 also stated that it will become an habit to use DevOps due to the benefits derived from it.

> *I work for an organisation that has adopted DevOps and therefore it is practiced for all our projects. However, if I was it another organisation that did not practice it, because of its benefit it will become an habit to use it for all the projects that I am involved in, provided the organisation is open mined to implement and support it* ***(Participant 3)***

Participant 2 said that due to DevOps's collaborative culture, it became a habit.

> *Yes, definitely. Now it became a habit to collaborate more and get others viewpoints when you run into difficulty* ***(Participant 2)***

Participant 1 also mentioned that collaborative culture became a habit with continuously improving processes.

> *Worked on few features using DevOps and it became a habit to collaborate with others and solve problems as a team with diverse skills and knowledge. It also became a habit for me to continuously see how I can improve the testing in DevOps for my team* ***(Participant 1)***

While Participant 4 said it became a habit because it is entrenched in their workplace practices.

> *DevOps is embedded in the way we work and therefore becomes part of our daily routine.*

Participants agreed that DevOps had become a habit in their working environment. This habit was developed due to the benefits, collaborative culture, and ongoing process improvement generated from DevOps.

Hedonic motivation

Hedonic Motivation refers to the enjoyment or pleasure of using a technological device. Table 5.10 illustrates the themes for this factor.

Table 5.10: Themes pertaining to the factor Hedonic Motivation

Factor	Themes
Hedonic Motivation	Job satisfaction
	Constant learning
	Project success

Participants 3 and 6 indicated that DevOps increased motivation since it improved the project's success and had fewer post-implementation issues.

> *You know working in software development, the most demotivating factor is when you spend so many hours working on a project, only to get a call late at night telling you there are problems with the software feature implemented. Now with the DevOps approach to things the motivation levels are higher because the likelihood of success is greater and you do not get crushed with comebacks as often as before.* ***(Participant 3)***

> *Before you felt like working in the dark, you have your fingers crossed that when it goes into production it will not fail. But now you feel more confident in the software being developed and you enjoy your work more.* ***(Participant 6)***

It has been reported by Participants 1 and 7 that individuals are more likely to communicate and work together, which ultimately leads to the team enjoying their successes and accepting responsibility for their failures. This encourages people to continue to follow the DevOps strategy.

> *People talk more, collaborate more and respect each other more and therefore it motivates me as an individual to go above the call of duty. There is no more them and us mentality. Now it's we. We all succeed together or we all fail together. When we fail, we all learn from the failure. Nothing is more motivating than having shared responsibility for success and failures* ***(Participant 1)***

> *The operation guy should always take the hit with a feature does not work in the operational environment. Something goes wrong and you contact the developer and his response is that I gave you a working feature and now I am too busy on another project to look at it. Previously it was the*

classic case of throwing the code to the operations department and not taking any responsibility thereafter. However now with greater teamwork, there is no blame game and this interaction is better and work more enjoyable ***(Participant 7)***

Participants 2 and 5 reported that working in a DevOps environment improved their job satisfaction, reduced their stress levels, and provided them with an atmosphere where they could continuously learn new tools and technologies, ultimately leading to them enjoying their working environment more.

DevOps definitely improved my job satisfaction. Being in security, you want to be consulted early on what security risks exist and how they can be prevented when developing the app. With DevOps you are part of the multidisciplinary team and we are able to identify and solve problems sooner. You enjoy your work more ***(Participant 2)***

I've been involved in the initially where our tools were the first to run on the new infrastructure. It wasn't automated with Chef. Then we started automating with Chef (I hate Chef... it's horrible) but the principle behind it, of creating scripts and automating things that you do... we should always be doing that. We should never do stuff, go and log into your Windows, and click because that's no on because if you want to do that again you've got to go and click. The learning around it, the levels of automating more and more so, you start with a little bit and you start incriminating that... it certainly made my work. I'm running 7 to 8 vendor provision tools. I write some utilities and things to help with my job, but I'm not a developer. With this DevOps journey, it certainly made it a lot interesting and I've got more stuff that I got to know and expand on, which for me it's a plus ***(Participant 8)***

So, when the bank, last year they did the 'Next-Gen' infrastructure, building an internal cloud and then there was another team that built 'Magic Dragon' or 'Puff,' which is an open sourced cloud. I had to go live but we ran out of infrastructure so, I built my stuff on both. I had to build it in this one, and then in that one. These guys were ready when I took it out but these guys weren't so, I went in there and now with AWS I've got my application running on the testing cloud infrastructure to evaluate it. All of that stuff, for me, is very interesting. It's good to grow and it adds facets to my job that keeps me here but whether I should be worrying with that stuff... that's a different question but yes, I'm enjoying it and I think it's definitely, we're going to be a lot closer to or a lot more flexible and whether that's because we're going to the cloud or not... I think that the whole DevOps journey has made things a lot interesting and a lot more resilient ***(Participant 5)***

I enjoy my job much better now than I did previously. There's far less stress, I think, because previously you were under tremendous stress. A nine-month project and you go live after nine-months and if things don't work... it's quite stressful. Whereas with the new ways of working its smaller chunks and things like simple, small things that you don't think will make a difference but it would make a huge difference is, for instance we're now… Previously, we were never allowed to put code that wasn't active into production. Whereas now, we are allowed to do that. So, we're allowed to put in stuff ahead of the time, stuff like that, which to me, makes a difference ***(Participant 4)***

I suppose one thing I could say and I always say that to the guys in my team, to the grads that we meet with as part of the graduate recruitment and so forth. Take this opportunity. It's such an exciting time in the industry and if DevOps is one of the mechanisms... embrace it because it gives you an opportunity to get involved in an amazing array of technology and it motivates you to learn more and be up-to-date with technologies ***(Participant 3)***

Participants agreed that they enjoyed using DevOps. Project success, collaboration, improved job satisfaction, reduced stress levels, and constant learning of new tools and technologies all contributed to this enjoyment.

Success Factors

Organisational Factors

Participants identified organisational aspects that are important for DevOps success. These are documented in Table 5.11.

Table 5.11: Themes pertaining to organisational factors

Factor	Themes
Organisational	Management support
	Communication and Collaboration
	Flattened organisational structure

Management support

DevOps teams needed to have the visibility of higher management since this ensured that management was involved in the process and was interested in its success. The following is an excerpt from Participants 1 and 8 that also emphasises this point.

> *Visibility is important there. If it is from the upper management support I think visibility would play a key... for my my view key aspect to say that we are part of the journey. Not just send out a comms. We are part of the journey. We see what you guys are going through and we know what needs to be done and we support that. Because to be honest teams are always led by a captain and it would be nice now and then that you can see your captain is also part and parcel of this journey and your captain understands where you guys are at* ***(Participant 1)***

> *Support. So executive support and sponsorship. If you kind of run a DevOps experiment within a organisation that doesn't have that sponsorship, you just get seen as that kind of, the cowboy team, the team that doesn't align to strategy and you get shunted. So if you don't have the support and if you, you'll be, you will find it very difficult to scale. You might be successful in a very small corner in a room in an office but trying to expand it you'll, you'll get fightback, right. Your governance teams won't align with you, they'll tell you that, you know this is not a thing* ***(Participant 8)***

It is also necessary to have top management support to guide teams through the decision-making process. As Participant 2 pointed out, for instance, when there is a disagreement over the tools and technologies that need to be adopted, top management can step in and make decisions that the team needs to implement.

> *Definitely the top management driver behind the process is very important, and the direction, in term of where we're heading, from an organisation point of view because the initial thing was we were all technically arguing around whether you Chef or this or that, or the next thing and everybody just want into circles. Until an executive said, 'You will use this.' Then we all were using this... some complained but at least everything moved forward. So, that executive drive and that is very important behind it. But to also have a clear direction in terms of what the objectives are. Like now, I think we've got a clear directive to go to public cloud and not just internal. We have to have to get that cost saving there and that forces things to open up. There're political pockets of people that disagree so, if you don't have that executive support you're never going to get beyond that argument in a room. You're going to have very expensive people disagreeing with each other doing nothing or you can have nobody doing nothing, it's the same thing* ***(Participant 2)***

For DevOps to be successful, management needs to be the one who facilitates change. Participants 2 and 5 mentioned that management must provide constant support for the strategy after it has been developed and put into action. This is necessary to ensure that all members of the organisation buy into this strategy.

I think it's absolutely pivotal. In something, as an organisation, which I believe we are also going through various iterations it's difficult to actual mature your DevOps teams or model in pockets if the organisational structure kind of goes against it because it feels like you're always swimming upstream. In the early stages it was very frustrating because, like I mentioned, certain teams were not on the same maturity level, or even on the same page to be quite honest with you. There were certain teams that were still feeling, I almost want to say, threatened that no, you're not allowed to look at this. You're not all allowed to have access. This is only me and my time. We used to have a separate team that looked after our middleware, our application service and our web service and that was a team that actually... we had to give our code to that team because they were the only ones with access, to actually deploy our code. But it used to be a constant issue because it was the salute, this one was blaming that one and that one was blaming this one. So, it was very frustrating and I believe that illustrates how important the executive support an organisational structure is to enable teams to adopt this model, which I believe our organisation has done and is continuing trying to define and improve on. ***(Participant 3)***

If you don't have support from the top that's your catalyst for change. Like you say, as we were discussing earlier, not everyone likes change. Unless someone weighs in on it that has authority everyone is going to stick to doing things the same way. Or drag their feet... this taking time is crazy so, unless you have some form of authority helping to drive the change and buying into that change, then it doesn't happen ***(Participant 5)***

Participants 5 and 6 suggested that management should also make time available to meet with teams to understand their challenges better and facilitate the speedy resolution of those challenges whenever possible.

We started on the Agile/DevOps journey, we were one of the first about two-years ago, and there we have scrum of scrums, which is run by the CIO. He needs to be there, the business is there, all the RTs, the stream leaves of the different groups or the stand-ups are there. They're given the necessary support. If there's anything that needs to be escalated it gets done either on the day or at that scrum-off-scrum session and we'll have a scrum-off-scrum for IT and then we'll have a scrum-off-scrum with IT and business together. So, there is definitely top down support ***(Participant 5)***

So you have things change at the top and how does that trickle down towards us and so on. So we have connect sessions. We have our executives coming through just to see how are process is. For us to understand what has changed so far. Are we still in line with our mandates? Are we still

going okay? How fast are the trains moving? How much of the things that we have already put into production, how much usage is ... how much usage do you have and how satisfied people are, customers are as well? So we do have those ***(Participant 6)***

Communication and Collaboration

Participants 3 and 6 mentioned that communication and teamwork are essential to the success of DevOps. Face-to-face interaction is required for this cooperation, and it is recommended that teams be co-located in the same building to converse with one another in real-time. Furthermore, when you are in contact with someone face to face, you can develop a stronger relationship with them.

Look, co-location is absolutely critical more than ever. It's about collaboration, cross-pollination, you talk about the X-type resources... it's very important. ***(Participant 3)***

The culture, yes we are still referring to them as business stakeholders. I don't know why because technically they are still part of the team. They product owners but they are part of the team. They form part of the framework. Because of that and because we are closely core located. A lot of the teams that we actually meet with other ops, things that we might meet with them in the entire process is all situated in one building. So if you are ... if you are going to send that mail, make sure you walk downstairs to make sure the person got it. Have those interactions, have those talks, how those daily stand-ups, have those retrofits, have your ... from time-to-time just go grab a coffee with the guy or the lady. From time-to-time make sure that they are also part of your stand-up. You know so those kind of things we practice a lot. So the core location is also a very important part of the culture portion. ***(Participant 6)***

Making sure that the people actually interact. Not say no we sent it through the mail, if you didn't see the mail it's your fault. And we try to refrain as much as possible from finger pointing because if a team sets out to deliver something then every single member in that team should give their 100%, should give their all. It makes it easier to pick out when someone isn't performing well and to correct it while still time. And if you can correct it, correct it and correct ways of thinking as well. So yes the culture of core location, your ***…(Participant 6)***

From an infrastructure point of view, we've worked like DevOps and Agile approach if we needed to sort something out... we've always been two space housing and security and also, the SDLC environment. You grab the right people together in a room quickly, sort out what you need to do, and off you go. You don't try and send off mails... you go to the person. So, that's just how we've always worked. ***(Participant 5)***

More flattened organisational structure

Participant 4 asserted that DevOps has evolved from a hierarchical structure in which developers' perspectives were not taken into serious consideration to a more flattened structure in which inputs from several disciplines are valued.

> *I also think that previously, the developer was at the bottom of a hierarchal type structure and the hierarchical people above didn't have a clue what they did, we were just managers or whatever, and the DevOps type methodology and Agile has brought the developer back more to the centre again. Developers more involved with the decisioning and what have you. Previously, it was the manager that made the decision whether he went live or not, and sometimes a manager would say, you will go live and a developer would say, 'no.' Then they go live and there's a disaster so, yes, I think that's a very big change that has actually changed the organisation of… Giving more power or autonomy to the developer and having the developer's input previously, you will deliver on this day, we are going live on this day, and that's it. It didn't matter what but the developer just had to attempt to deliver. It's much more, better* ***(Participant 4)***

People

Participants identified people and team aspects that are important for DevOps success. Table 5.12 illustrates the themes for this factor.

Table 5.12: Themes pertaining to the people factor

Factor	Themes
People	Competent
	Adaptable
	Motivated individuals
	Autonomous team and cohesive work unit

Competent

Participants 1, 4, and 6 all agreed that having competent team members is critical to the success of DevOps. DevOps teams require members who possess the appropriate abilities and are eager to grow their expertise by learning from the other team members.

Must have the right people. To be honest the right people and the right skill. And again it should even if it is the right people and the right skill you also need to learn from each and every iteration, each and every sprint, each and every iteration that you go through. It is important that those ceremonies allows everybody to have a voice in what they do. The moment where one team member doesn't have a voice it is not a team discussion. For me then it is not a DevOp team whatsoever ***(Participant 1)***

It is the future. It is the future but it should not be the silver bullet. It shouldn't be the silver bullet. At the end of the day it is just a process, it is a methodology. It is the people that make it work and if you don't have the right people with the appropriate skills or you not looking after your people, you not getting your people involved into what you are trying to sell them it is not going to work ***(Participant 4)***

That is hugely important. People's thoughts processes. If you have the right people thinking the right way and willing to hear out the other person, then you can easily form a great team of great teams that not only follows DevOps but also has important skills. So the people portion is very important ***(Participant 6)***

Adaptable

People in DevOps need to be agile and must be able to change and embrace DevOps in order for it to be successful. Any team member not embracing the change that DevOps brings about will make DevOps unsuccessful, as reported by Participants 1 and 4.

... it just really talks about the ciaos bringing in new ways of thinking, new ways of working to an organisation that is stuck in a specific way. And that is what methodology does is that some people are stuck in one way they don't want to change and you need to embrace it and need to be able to understand. There is a reason these new things are coming in so you can achieve a certain outcome. If you can't embrace it guess what you will be that kink in that chain that will break everything ***(Participant 1)***

So, if you have a team that is willing to accept change then I think this process would be a lot easier. But if you have people that resist then there's only so much that someone can do, right ***(Participant 4)***

Motivated individuals

Participant 2 also mentioned that DevOps ought to have people who are not only motivated to find solutions to problems but also motivated and passionate about working with others to accomplish the tasks.

Well, you need to have passionate people around. So, we get a lot of documentation and design and all of that but you need to have people that wants to sit down and solve the problems to move forward. So, you have your journey but to work out the technical implementation and then find those guys and then when those guys start talking to each other then things start happening ***(Participant 2)***

Autonomous team and cohesive work unit

Participants 1 and 5 agreed that DevOps allows teams to make their own decisions and allows all team members to provide input during a project. In addition, these teams function cohesively, making it possible for team members to exchange their knowledge with one another effortlessly.

That is where it plays as well is if it is a highly effective team that anybody in that team should be able to put his or her hand in and lead a discussion or lead a stand up. Should be just Modabi that does the speaking. Shouldn't be just the RTE. Maybe the development must also lead the discussion. Maybe the business person must also lead the stand-ups as well. To be able to create that autonomous team to be able to create a team where anybody can now say okay fine I'm leading the discussion today. This is a topic and this is the issue we are having and this is how we need to deal with it. Shouldn't just wait for okay because you are RTE you should always lead the discussion. We need to be able to change those roles. Just you know so that you can see everybody is on equal standing. ***(Participant 1)***

I've seen a significant skill shift with the guys sharing knowledge now, from a development and around the bank perspective. So, that knowledge transfer is lifting the skills of a lot of teams. Not just within individual teams but having more interaction with security, with your infrastructure resources. It's actually bringing that knowledge into the whole development process so, that's been good ***(Participant 5)***

Tools and Technologies

Participants detailed the importance of tools and technologies for DevOps success. Table 5.13 illustrates the themes for this factor.

Table 5.13: Themes pertaining to tools and technologies

Factor	Themes
Tools and Technologies	Selecting the right tools
	Decrease complexity
	Variety of tools
	Infrastructure provisioning and automation
	Support various parts of the DevOps cycle

Selecting the right tools

Participant 1 suggested that for DevOps to be successful, the appropriate tools are required. The DevOps process can be negatively impacted, decreasing its effectiveness and efficiency, if the appropriate technologies are not used.

> *You must have the right tools. Some... the right tools for the technology that you are working on. Technology should be your friend. Tools should be your friend as well. So often if we don't have the right tools to deliver something faster you would end up... you would end up just ruining the whole methodology process you want to do. So technology and the people as well as the risk play an important role* ***(Participant 1)***

Tools can decrease complexity

The use of tools has the potential to reduce the number of manual operations that are required to take place during the development and deployment of software. Participant 1 offers the example of testing to illustrate how using tools has made testing procedures more effective. Because of this, it was also possible to test a more significant number of scenarios, which ultimately led to an increased possibility that the program would be successful in the operating environment.

> *So from a testing perspective and again there should be where it is not just testing it should be it is Dev and testing together. If I look at the tools that we using from a mobile perspective, the cloud tools for example AWS it plays a... I have to say it a very, very critical role in terms of how we need to cover a specific handset. So now you get a tool that has millions of phones. Not emulators real devices where instead of where you able to*

test on a Samsung, I need to check if it works on an IOS device and this is manual things that you are doing. Where now you can actually just configure your scripts and it pushes it out to the cloud and they test on all those basis for you and all you need to do is from a customer perspective say right if I was a customer is this thing able to do that. Then you focus on actually the functionality and not the compatibility because the tool is sorting out the compatibility for you. Especially with cross browser testing as well. You get your cross browser tools and recently one that we got was Smart Bed where you run the same test case and it tests on different browsers for you. Your IEs, which would have been very labour intensive if somebody had to go through it manually. Now you are able to test it once. These tools record what you are doing for you and then it tests on the different browsers ***(Participant 1)***

<u>*Using a variety of tools*</u>

Participant 2 mentioned that various tools can support every stage of the DevOps lifecycle. Participant 2 further suggested that a particular tool used by one team or project may not be suitable for another team or project. For instance, Chef allows for infrastructure provisioning to enable continuous software delivery, similar to Ansible. However, some developers like using Ansible because Chef does not sufficiently provide all of the team's requirements.

We're in the tools team so, initially when we started in our area is we provided looking at the Agile development methodology so, we provided Jira, which is a task management system and it's got an agile plug-in so, it's got the scrum and all of that. Together with that it's got confluence, which you actually do your documentation designs and all of that... the storing on that. So, that's the one part of the agile journey and then another part of agile is to be able to continuously develop and deploy and test, and that's where we host Bitbucket for the developers so, every time we check in your code into Bitbucket then it kicks off a build process on Bamboo and then Bamboo in turn, would then compile whether its Java, or web, or whatever software... it compiles it into artefacts and then it also, added the automated testing components to that. Then also, at the point when you now package it, it also provides a distributional implementation of that. So, that's where we started. Then also, I don't know if you know what Nexus is but the built artefacts get stored in Nexus so, we can share it across teams or be deployed from Nexus. So, we both store the source and the built application. Then when the DevOps journey started about four or five-years ago, the need was seen to take, because we use Bamboo to automated the packaging of the application and then also deploying it into that. Then we added the capability to actually provision the infrastructure to do that thing so, the DevOps journey was started and then Bamboo was one of the tools that they used to do some of the deployments and kick-offs. ***(Participant 2)***

So, the directorate is use Chef, and we've done a lot of that, but Chef has problems and it doesn't cater for all of it. So, some of the guys are using Ansible and some of them are using some other technologies to do that provisioning but that part only allows for the configuration of the machine. ***(Participant 2)***

Tools for infrastructure provisioning and automation

Participants 2 and 3 have expressed their belief that tools are crucial in infrastructure provisioning and automation. When these tools are unavailable, it will be difficult to provide infrastructure on time and develop platforms that support continuous integration and deployment. This collection of tools needs to be mixed and matched according to the problem being solved.

So, in terms of tooling, definitely the DevOps is to automate your infrastructure provisioning. Without tools you can't do that so, yes, tooling is definitely an enabler for it. The choice of tooling is not as important as the process that you're solving. I don't have any holy cows, whether I use Bamboo or Jenkins... it's not as important as the script to write to provision the infrastructure. Now, whether that script is in Bash, or Chef... as long as it's understandable and sharable, that's what it's about. ***(Participant 2)***

I want to turn this on its head and yes, it's absolutely critical but for me, we've reached critical mass when we entrusted our engineers, we gave them the freedom to use what they needed to use. It's very difficult for me to mention specific tool sets because what we use today might not be what we use tomorrow, but that's the key. The guys are fully empowered and intrusted to make use of whatever toolset they need to make use of in order to solve the problem and it doesn't have to be a 'one size fits all.' It never is 'one size fits all.' But yes, by all means there is a set of tools that has made our life much easier. Just looking at the stack and how that helps us in terms of our new ways of work methodology. You can talk about all the CICD pipeline product sets that's available, both on the distributor systems and the mainframe. Absolutely, all those things are vehicles that helps us drive or helps us along the journey but we are not confined to a specific product set, but we've got the freedom to use what we need to use. ***(Participant 3)***

Tools support various parts of the DevOps cycle

Participant 5 mentioned that the DevOps process uses various technologies to help support different parts of the process. Jenkins, for instance, is utilised for continuous integration and delivery, Splunk is utilised for post-implementation software monitoring, and Expeditor is utilised for software testing.

> *Yes, the right tools. So, we use a lot of the compute products on the mainframe. We use ISPW for our source code management. Then that interacts with a lot of our testing tools and data management products. We had all these tools before and now we have a workbench that integrates all these tools together. We use Expeditor for testing and it's also a debugging tool, and what we have, because we're familiar with all these tools when we want to bring something like Total Test online, which allows us to record our debugging sessions, it's now a consolidated tool, we're already familiar with it so, it's quicker to adopt to neutrals. Then, also from a Compuware perspective they've now bought in to dealing with a lot of the open source players. They will be able to integrate with Jenkins, which then will allow us to coordinate releases with Open Systems, because they can talk to the same tools. So, while we focus more on what we're doing on the mainframe for now, how we can start relating to Open Systems... we've already opened that doorway so, we can grow into that. We're not there yet but we can certainly grow there. Our monitoring capability is completely separate though so, that's still separate toolsets. So, where monitoring is the critical part of DevOps as well, that's very separate from our deploy capability, at the moment... it's a separate toolset. For monitoring we use VMC Main View and then we also recently started feeding data in, which we've just reformatted our logs in Splunk and that type of thing, and get more visual representation on our data. That monitoring you'll see if anything changed compared to what your last release version was. So, it's similar to the Chef that we use in the card space where we've got a compliance component part of that as well. Where it will actually run and check your compliance and look at the last bill and if anything has changed... actually flag it. You can actually set it that it automatically reverts back to the original version.. **(Participant 5)***

Automation

Participants explained the role of automation in DevOps success from their experience. Table 5.14 illustrates the themes for this factor.

Table 5.14: Themes pertaining to tools and technologies

Factor	Themes
Automation	Improves speed and quality
	Enables automated testing
	Create self-service infrastructure
	Deployment of code
	Support the entire pipeline

Automation improves speed and quality

Participants 1, 6, and 9 mentioned that automation helps to speed up the activities involved in the pipeline for software development and deployment. Not only did the speed increase, but the procedures also become substantially better. Better quality is achieved throughout the entirety of the software development pipeline as a consequence of this.

> *So automation it's certainly improved the processes. It might make certain functions or roles redundant. But it would then open up opportunity for other focus points, without a doubt* ***(Participant 9)****.*
> *I have to say that automation also plays a very significant role in getting us to get things in faster. But automation only works if your code is stable. Especially in my environment if you got a stable code base where Dev has done its work in terms of they've actually automated their integration testing as well. It just makes it so much easier when we press the button and the thing works. (Participant 1)*
> *I know for a fact the moment I started practising SaFe I need to start looking into DevOps. I need to start automating my tasks and at the end of the day I need to deliver quality and I need to deliver quality more frequently. What helps me deliver quality more frequent, might not necessarily be manual processes but a lot of automation where I can. So if you can automate a lot of the processes then you have to go for it.* ***(Participant 6)***

Automation enables automated testing

The process of testing software by utilising pre-existing test cases without the need for human participation is referred to as automated testing. As mentioned by Participant 3, this automated method of testing frees the developer from the burden of performing manual testing, allowing them to focus on other aspects of the development process. Continuous testing is one of the most essential practices within DevOps, and automated testing makes it possible to conduct it.

> *Absolutely critical. Again, think of automation almost as a carrot. I don't like referring to people as developers anymore because we've evolved way beyond developers. We are engineers but in the old realm of developers and ops... automation is that carrot for an ex-developer to say, 'what, you want me to now support this thing in the middle…?' How am I ever going to get any work done if I need to do production support? You're going to programmatically solve it, that's how. Let me give you another example. A couple of years ago, when we transitioned into feature teams and so forth. It so happened that one of my teams were component teams and the makeup of a component team didn't allow for testers within my component team. So, we were a bunch of developers and then we would feed into various feature teams, and feature teams would have feature analyst, testers and so forth. I remember the developers calling me aside and say, 'this is ludicrous... I've been doing this for 25-years, with all due respect my salary is X-amount per hour and you want to waste money and let me test?' I said, 'Yes, how can you not test?' No, this is absolutely unheard, no ways. I said, 'I'll tell you guys what, I have X-amount of positions available. If you guys feel so strongly that you need a dedicated tester, then which of the developers are we going to sacrifice because that's the only way we're going to get testers?' What's more important? Do you want developers or testers? End of discussion They automated their testing... automation is key.* ***(Participant 3)***

<u>*Automatically create self-service infrastructure*</u>

In the past, when infrastructure was needed, the operations teams would manually configure the servers. The fulfilment of a request took a considerable amount of time since manual intervention was required in order to do so. The manual configuration of the servers also led to a lack of consistency in the configuration of the servers, which ultimately resulted in the servers being taken offline. Infrastructure as code (IAC) was utilised to generate servers in an automated, rapid, and consistent manner to solve this challenge. The following participants, numbered 2, 5, and 8, present evidence that supports this claim.

> *Automation, the ability to scale . To be able to sustain that sort of demand is self-service with automation. So traditionally in organisations like us if you wanted a server, you log a ticket. If you wanted somebody to look at*

your server and what's going on you log another, you log a ticket and you get into the queue, alright. Today, certain, certain platforms, we allow developers to build their own servers, that's why I kind of mentioned, you can build servers in two to three to five minutes. They, they are preallocated a certain resource pool and they can build against that and that's how we work. So I probably think that self-service that that's probably the most important part of it. ***(Participant 8)***

Four years ago, if you wanted a server it could, it took about four to six to eight weeks to get a, a bit-off server for example. If you knew somebody, maybe a week, if you really knew somebody, maybe a day or two, okay. Today you can get one in between two and ten minutes depending on what server you need, okay. As long as it's a strategic server that we need. We've put, we've invested a lot of effort into the automation of that capability to give you that sense. ***(Participant 5)***

So, for me, using automation enables you to consistently create your environment because every time that you deploy or make enhancements to your application. In the past, there'd be little things changed on the server and you don't always know that and then your production gets depleted. So, your infrastructure is not necessarily consistent and it becomes almost like a old cow. It's like that is the server and you can't log into this thing because you can break this and that. Where now, if you have a pre-defined server or servers because it'll be the application on the database with however many different parts you have to be able to create each one of them consistently. ***(Participant 2)***

Automatic deployment of code

Software is built, tested, packaged, and deployed automatically through automatic deployment, eliminating the need for human intervention. As a consequence, the software can be deployed into the operating environment at a faster rate. Containerisation is a method that allows for the deployment of features into the operational environment to take place within containers. Deploying features within containers reduces downtime because, in the event that any changes are required, they will only affect the container that requires the change. On the other hand, all other containers will remain unaffected, which results in increased service uptime. Within the context of DevOps, participants 2 and 7 highlighted the use of containers and automatic deployment of code.

we use sonar for static code analysis. In Bitbucket you can enforce that you can't check in your code unless the build is clear and all the tests are passed, and a peer has reviewed it. So, the quality and that is all, and then the DevOps part is adding that automatic deployment. Before the automatic deployment was targeted at getting your code into an

environment, and now, with DevOps, you've added the thing of actually creating an environment and deploying your code into the environment. So, creating and/or sustaining and maintaining the target environment to a consistent state. Nobody should log onto that machine and install something or change something for the app to run. It should be all here from the script because anybody can run that script, and it will work. ***(Participant 2)***

containerised applications that get deployed, if something goes wrong or the node fails, it spins up on a different node in a different part on Kubernetes or OpenShift and it's the same. The container that fails does not affect other containers and is restarted immediately to prevent downtime. You've also got AWS and Microsoft Azure, they run a lot of their stuff on containerised platforms. ***(Participant 7)***

Support the entire pipeline

According to Participant 5, automation can be deployed throughout the entirety of the DevOps pipeline from start to finish. This makes it possible to reap the benefits of automation throughout the entire pipeline, which ultimately results in more rapid and consistent procedures.

We can't do it without automation. There's also differing levels of automation so, there's our deliver and deploy capabilities, which we've automated via our ISPW. There're also our testing and monitoring capabilities. If you're on the mainframe you're automatically monitored. Automating the entire pipeline provides additional value and consistency. ***(Participant 5)***

Culture

Participants explained from their experience the importance of culture in DevOps success. Table 5.15 illustrates the themes for this factor.

Table 5.15: Themes pertaining to culture

Factor	Themes
Culture	Collaborative working culture
	Shared responsibility
	Continuous improvement
	Culture of automation

Collaborative working culture

The combination of professionals from development and operations is what constitutes DevOps. Since each individual possesses their domain knowledge and objectives, conflicts may arise. As a result, Participants 1, 3, and 4 indicated that for projects to be successful, all require a culture of working together in the DevOps environment. Everyone is accountable for finding solutions to problems that arise during the development and deployment processes. Participant 5 discussed how a culture of collaboration leads to knowledge sharing.

> *I can only speak in terms of projects that I've been dealing with and again it's... it is sometimes 50/50 and sometimes 60/40 or 70/30 is that if even the individuals that are leading the changes are not living the culture of what we are trying to do from a Dev perspective you will definitely have failure. That you do not pick on one team over another team. Everybody you know if you working in a DevOp organisation everybody should be held at the same level in terms of responsibility, in terms of seniority, in terms of delivery, we shouldn't be creating hierarchies and I think that is what the whole DevOp thing is trying to get rid of creating hierarchies because we need to live as a team. Not as a him and her she or they. It shouldn't be you know it's the development's problem. So if I have a defect for example it shouldn't be I'm sending the defect to the Dev and saying ah meeting you with the finger because you called it badly, you need to fix it and I'm not going to allow you to go into productions. Should be a thing where if we find a problem we deal with it together and we get the resolution together so I am able to assist much faster. Not just throw it over the fence or saying ah no you know what it doesn't work. We will put it in production and then we will sort it* ***(Participant 1)***

> *It's all about culture. Culture eats strategy for breakfast.. Again I know we're not a Google, we're not a Facebook but if you walk into a Google or Facebook and you look at how the setup is, it doesn't look like a corporate. It doesn't look like a corporate yet they are churning billions if not trillions of dollars every year because of the culture. The culture was set up in a way to say for us to succeed we need to be able to do XYZ. Not for us to succeed we need to make sure that that thing goes in on that date. They are successful because they've built a strong culture where they are saying for us to succeed everybody needs to be onboard. Not just one person needs to be onboard. Everybody needs to be onboard and you can see it even from there. From the exec know what is happening in the organisation as such as well.* ***(Participant 4)***

> *Culture, absolutely culture. A couple of years ago, two or three years ago, I had a developer in the COBOL space in a room like this and we were*

going through a demo of a new product tool set and the ability it gives you to actually publish your services, internal or external, how do you get people to subscribe to your API? How do you manage the throttling and the performance thereof, and so forth? She said something that I will never forget, she said, 'I don't care about any of that... I just want to write my code.' I couldn't believe it and I said, 'sorry guys, but there's no space for that.' You cannot tell me that you do not care about that. It's not somebody else's problem... it's our problem. So, yes, that example comes to mind. ***(Participant 3)***

I've seen a significant skill shift with the guys sharing knowledge now, from a development and around the bank perspective. So, that knowledge transfer is lifting the skills of a lot of teams. Not just within individual teams but having more interaction with security, with your infrastructure resources. It's actually bringing that knowledge into the whole development process so, that's been good. ***(Participant 5)***

<u>*Shared responsibility*</u>

Participants 3 and 8 said it is essential to function successfully as a cohesive team. Whether the deployment was successful or unsuccessful, the team accepts responsibility for both outcomes. If there is an issue, it must not be left to the discretion of any one person to fix; instead, it must be resolved as a team effort.

I'm very blessed and I'm so glad I got involved in all the different technologies, and call it both worlds, both sides of the fence. I remember when I was in infrastructure and I didn't realise it back then but looking back at me now, I realise how I was and I recognise it in the people around me. I had a very narrow outlook on the world. It was my database. If my database is running optimally and there's no errors, etc., then sorry, I can't help you, go away. Or it was my OS. Exactly, and the same now on the other side of the fence from a software development perspective. It's as long as my code, as long as the quality is great and as long as the functional requirements are signed off... I've got the best code. Guess what? It means nothing. Is it adding value to a customer? If the customer is unable to actually get a login page then whose problem is it? The short answer is, it's everybody's problem. I think the definition of done is very important for every team. Define what 'done' means to you. Done, in my mind, should be as soon as it's a production... then it's done. we're fighting that culture. I've got developers that are telling me, as soon as the code is in SIT then it's not their problem. Then it's the testers problem and that might also need to change. Which is again, why I hate still referring to testers and developers or ops support or operations. It's engineers. So, the titles must be done away with, basically? As a team, you just say, 'who is capable of doing it and continue as one team.' ***(Participant 3)***

Team culture is important especially when a feature is deployed. Everyone take credit for it success and everyone needs to fix it if it fails. The team builds it, the team needs to own it. It must be a collective effort by all team members to solve problems and not blame any individuals. ***(Participant 8).***

Continuous improvement

Participant 9 stated that both individuals and teams must engage in active learning. Individuals are capable of learning on their own as well as at the organisational level. Because of this continuous learning, the teams will be exposed to new tools, which will improve their delivery speed and quality.

So I'm driving a continuous delivery Dojo at the end of October which aims to actually bridge the gaps around the DevOps culture and transformation, okay. And we've started with Microsoft using their version of DevOps platform which is quite exciting because there's a lot of teams that have registered. We've got a full house which is probably going to be the biggest live hack with Microsoft. From a customer perspective which aims to do two things, it aims to create a learning experience for our teams that are traditionally Legacy based and grow their awareness in terms of the life cycle of automation and continuous delivery. And then also help them build with hands on experience. I think the self-learning thing is one thing but being able to apply it simultaneously in a collaborative safe space, I think would add a lot more value. So we've come up with this Dojo. We've done Dojos in, in the organisation before, very team specific, over a longer period of time to basically drive that transformation team for team. I think for us to be able to reach a DevOps state in its true form, we went for a Dojo at scale. So we're going to be driving a whole lot of these initiatives from a learning experience and then building experience as a weeklong block and leveraging strategic partnerships with the likes of Microsoft or AWS or whoever else we deem fit. And then obviously also with our tool prevent, vendors which is anybody from Jenkins to Cloud B's or the Cloud B's runs Jenkins and Sonar Cube, so that's Compuware and a whole bunch of others. And see if we can then partner with all of these actual software tools providers, to actually grow that DevOps understanding and what each tool set does for them. ***(Participant 9)***

Culture of automation

Participant 3 suggested that it is advisable to automate as much as possible. Nevertheless, it is also mentioned that before automating, you should clearly understand the reason for automating and the positive effect it will have on the operation of the pipeline. In addition, it

is essential to have a good understanding of the mechanics that underlie automation to be able to handle problems if something breaks.

> *as a unique software engineer or as a DBA you tried to automate as much as possible because economies of scale, you were a team of six or five DBAs supporting over a thousand databases, on the Continent, or a team of seven unique software engineers supporting 2.000 unique systems. You don't achieve that without a level of automation and we did very well. My problem is, and now I'm going to sound like a more senior veteran, but the problem is as the new kids came in, they found that level of automation without really understanding what happens behind it. It just 'poof' and magically it appears. But they don't really understand what's the science behind it. The people that knew the science moved on. So, now you're left with these kids that basically are just, with respect, because that's where I started my journey almost 20-years ago, as an operator. But they're basically operators. Now, all of a sudden, you say from tomorrow you will be an engineer and you now need to automate your infrastructure stack. Typically, a lot of our guys today, they can't even do normal bash scripting because they found it already there. Sorry, I made a long story about it, but skill and culture.* ***(Participant 3)***

Project type

When it comes to software development, there are many different process models. Each possesses its own set of qualities that best suit a particular project type. Table 5.16 illustrates the themes regarding the type of project suited to DevOps.

Table 5.16: Themes pertaining to project type

Factor	Themes
Project type	Business critical software
	Shorter time frame
	Emerging requirements

Participants 1, 8, and 4 have said that DevOps is suitable for projects that require a rapid turnaround, software that is essential to the operation of the business, and software that provides the organisation with benefits over its competitors.

> *DevOps is not suited for all projects. It is ideal for projects that the client needs quickly so that client can gain value as soon as possible.* ***(Participant 1)***

So, yah, business-critical software. DevOps is most appropriate for projects that are customer-facing. You want as little downtime as possible with customer-facing applications. DevOps is, therefore ideal for these applications because you want them deployed quickly and problems resolved with them as soon as possible. ***(Participant 8)***

Projects giving the organisation a competitive advantage are ideal for DevOps since organisations want the software delivered and deployed rapidly. Furthermore, using DevOps the project quality is better and less prone to errors in the operational environment. ***(Participant 4)***

Participant 3 referred to the fact that DevOps is best suited for initiatives with a shorter schedule. This is in line with the primary advantage of DevOps, which is to rapidly deploy software into an operational environment and continuously integrate new features into the one already in place.

Projects with a shorter timescale that can be broken into features which are built iteratively. ***(Participant 3)***

Organisations are consistently working to enhance both the processes they use and the products they provide to customers. The consequence of this is a continuous flow of new requirements that need to be developed and implemented. Participant 6 mentioned that the DevOps approach can do this for enterprises.

In the corporate world, especially banks, there is constant innovation and process improvements that are occurring. This results in new requirements always emerging. These projects are suited for DevOps which allows for the requirements to be developed and deployed quickly. ***(Participant 6)***

Process

Participants implied that following a DevOps process is essential for DevOps success. These themes are documented in Table 5.17.

Table 5.17: Themes pertaining to DevOps process

Factor	Themes
Process	Planning and management
	Continuous principles and practices

Planning and management

Every project requires careful planning, careful monitoring of its progress, and careful identification of any potential risks that could hinder the team from accomplishing the project's goals and objectives. Participant 1 emphasised the importance of following the DevOps process to implement project management principles successfully. In addition, Participant 5 highlighted the significance of DevOps methods in order to ensure that the project was not characterised by widespread chaos.

> *following a DevOps process is important because it establishes what needs to be achieved, how far you are from achieving it and whether are there any obstacles you will be facing further down the process cycle that you need to mitigate now. So, there is some structure in the entire process.* ***(Participant 1)***

> *The DevOps processes are important to the success of DevOps. Just like any methodology or framework, there are processes embedded so they can be successful when implemented. If we do not have these processes there can be chaos within the project because everyone will be doing their own thing. However, still got a lot to do in getting these processes fine-tuned using automation, etc.* ***(Participant 5)***

Continuous principles and practices

Continuous delivery, deployment, testing, and monitoring are all essential principles and practices incorporated into the DevOps model. Participants 8 and 3 indicated that by adhering to the DevOps process, projects would be able to obtain the greatest possible benefits from the principles and practices, increasing the likelihood of the completed project being successful.

> *Some important principles and practices in the DevOps framework exist, such as continuous delivery, deployment, testing and monitoring. These principles and practices can only be implemented within the processes in the DevOps lifecycle. If these processes are not implemented then the core practices of DevOps will be missing and the benefits of DevOps will not be attained. Therefore it is important to follow the DevOps process to achieve project success.* ***(Participant 8)***

> *Agile already incorporates software development and delivery processes. Operations handle the deployment and monitoring. With the merging of development and operations, all these processes are now incorporated under one umbrella referred to as DevOps. To achieve the benefits of*

DevOps I feel that these processes must be followed and we do follow them in our projects. ***(Participant 3)***

New factors

After a comprehensive examination of the transcription data, two additional themes, security and cloud computing, emerged that influenced DevOps' success. Each one of them is discussed in the sections that follow:

Security

As indicated in Table 5.18, participants mentioned that security should be incorporated into the DevOps pipeline.

Table 5.18: Themes pertaining to security

Factor	Themes
Security	Incorporated into DevOps pipeline

Incorporating security into the DevOps pipeline

Rather than being an afterthought in the DevOps process, security should be incorporated as an integral element of the DevOps pipeline right from the beginning. This is particularly important in the context of DevOps, which aims to produce software in a short amount of time. Participant 2 stressed that if it is not a part of the lifecycle, it is likely to be ignored because developers will be concentrating on attaining rapid deployment. From participants 3 and 5 experience, they indicated that projects that incorporate security from the planning phase have a more significant likelihood of having a positive outcome in terms of security.

Since DevOps produce features at a rapid speed, developers can lose sight of the security aspects that need to be considered and therefore it is important that security be in all phases of the lifecycle, which means that a security personnel need to be part of the team. Sometimes, this may not be possible since the number of development teams in an organisation significantly outnumbers security specialists but it is important that feedback flows between them. ***(Participant 2)***

The only thing I can add is the process we started with. The last thing we'll take away from that is security was involved from inception. Whereas, we've been part of other projects but only much further down the line, and literally at the 11th hour, and you're expected to pull a rabbit out of the

hat because they're going live that weekend, but you were never part of the design or initial meetings, etc., whereas this, we were part and parcel from the beginning to end and it makes a big difference. ***(Participant 5)***

You find now, on those newer projects we tend to phone security right up front to say, 'guys, this is what we're looking at doing... are there any things we need to take into consideration in our planning process? We get that advice whereas, trying to bolt it on and security should never be bolted on... it should be part of the build. ***(Participant 3)***

Cloud computing

Participants indicated that cloud computing plays a essential role in DevOps. The identified themes indicating this is in Table 5.19

Table 5.19: Themes pertaining to cloud computing

Factor	Themes
Cloud computing	Easier deployment
	Improved testing
	Easier provision of infrastructure

Easier deployment

The cloud infrastructure has automated processes that DevOps teams can use. According to Participant 4, this makes it possible for applications to be hosted in a timely and consistent manner, and it also has the potential to reduce organisations' reliance on their operations teams in the future. In their discussion, participant 6 said that software deployment is made more seamless by using automation on the cloud.

To be able to automate it... it's definitely going to grow. It's a good practice. There's definitely a lot of advantages around the automation aspects of it. Then that consistency that automation brings you. Yes, it just has to grow. I don't think that once you start working on DevOps way that you'll actually… If you think about it right now, we say our infrastructure

is part of our developing thing but if I look at the cloud services where they've got functionalities where you can just run your app. You don't worry about the infrastructure so, that might be the next iteration is that you just say, I run an app and its running in this container, or this Tomcat, or Docker or whatever it is and it just runs. You don't even care about the infrastructure. So, that's maybe the Dev-plus or the Dev-minus Ops or

something, which could be the next iteration of it. But I think that's probably where it's going to end up is that the Ops is not going to feature. The infrastructure is going to be gone. ***(Participant 2)***

Cloud computing is huge and it's the buzzword and it makes sense in so many ways, it's not even funny. The we have or the level of automation and maturity that we've achieved in some of our teams already. Whether it runs public cloud or internal cloud... it's irrelevant. Everything is code and Agile aside, the DevOps culture and mindset in terms of the full-stack engineering methodology, automation... yes, that just makes the transition to cloud so much more seamless. ***(Participant 3)***

Improved testing

According to Participant 1, cloud infrastructure offers testing methods that are both efficient and effective for mobile application development. Using the cloud, the developer could test the functionality of mobile applications without worrying about compatibility concerns associated with using various browsers and operating systems.

If I look at the tools that we using from a mobile perspective, the cloud tools for example AWS it plays a... I have to say it a very, very critical role in terms of how we need to cover a specific handset. So now you get a tool that has millions of phones. Not emulators real devices where instead of where you able to test on a Samsung, I need to check if it works on an IOS device and this is manual things that you are doing. Where now you can actually just configure your scripts and it pushes it out to the cloud and they test on all those basis for you and all you need to do is from a customer perspective say right if I was a customer is this thing able to do that. Then you focus on actually the functionality and not the compatibility because the tool is sorting out the compatibility for you. Especially with cross browser testing as well. ***(Participant 1)***

Easier provision of infrastructure

The challenge of providing developers with quick and consistent infrastructure for them to deploy their software has always been present. Participants 2 and 9 said that infrastructure can be created through the cloud, enabling developers to release products rapidly without worrying about the system breaking down.

So, ultimately, infrastructure itself must become a lot smaller and at the moment, it's still a big thing, especially if you go to the cloud but maybe that's more internal versus external cloud... do you want to manage your own cloud or not? The DevOps part should be the developers having that

infrastructure leg to be able to deploy quickly. So, in terms of that first part around the success factors from a Dev team... can they provision their stuff from an automated station and I think we're getting close to that using cloud. ***(Participant 2)***

In terms of a DevOps team like the DevOps guys that are sitting on the fifth floor that's looking at AWS and what it means. For them, the success is to be able to have the teams have as much freedom to be able to create the environments that they need, without much intervention from them that they can basically manage the process, I mean the actual service provider, the infrastructure with it, that is basically to be able to consume those DevOps' requirements from the teams as easily as possible and without letting the team break everything else. ***(Participant 9)***

5.3 THEMATIC MIND MAPS OF FINDINGS

A mind map is a diagrammatic representation of concepts and ideas connected to and organised around a primary concept or idea. Branches in the diagram represent major themes around the primary concept (Burgess-Allen & Owen-Smith, 2010). The thematic results of the qualitative analysis presented thus far were narrative descriptions. Figures 5.5 and 5.6 represent mind maps that encompasses all the themes discussed so far into a diagram that lets the reader immediately pinpoint the significant themes. Figure 5.5 highlights the themes of professionals' definition of DevOps, DevOps strengths, and its adoption challenges. Evident from this figure is that the definition provided by professionals comprised of many critical terminologies aligned to DevOps, such as automation, code ownership, culture of collaboration, people, process and practices and use of tools. Furthermore, participants indicated various strengths of adopting DevOps such as faster delivery, better knowledge sharing, ownership and accountability, improved product quality, problems resolved earlier and greater visibility. However, participants also indicated challenges to DevOps implementation, such as legacy code, not a one size fit all process, focus on delivery, automated testing and cost. Factors affecting DevOps acceptance are depicted in Figure 5.6. These qualitative findings helped validate the initial conceptual model established in Chapter 3 by illustrating from the lived experience of professionals that performance expectancy, effort expectancy, social influence, facilitating conditions, habit and hedonic motivation are important acceptance factors in DevOps. Furthermore, DevOps's essential success factors are automation, technologies, culture, people, project, process, and organisational factors. Further analysis also revealed security and cloud as two potential additional factors for DevOps success, with new

relationships established between automation and performance expectancy, automation and effort expectancy, technologies and performance expectancy, and technologies and effort expectancy. The preceding claim pertaining to the relationships that have been established is corroborated by running queries that identified joint references to the following constructs/codes:

- Automation vs Performance Expectancy – as illustrated in Figure 5.2, 5 of the 8 participants made joint references to both these constructs in close proximity of each other:

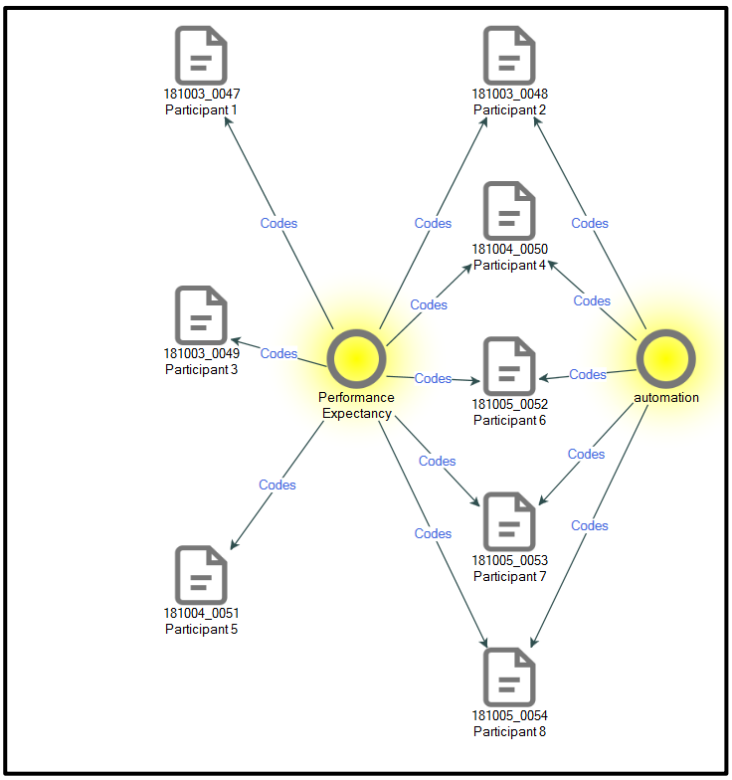

Figure 5.2: Joint References to Automation and Performance Expectancy

- Automation vs Effort Expectancy – as illustrated in Figure 5.3, 4 of the 8 participants made joint references to both these constructs in close proximity of each other:

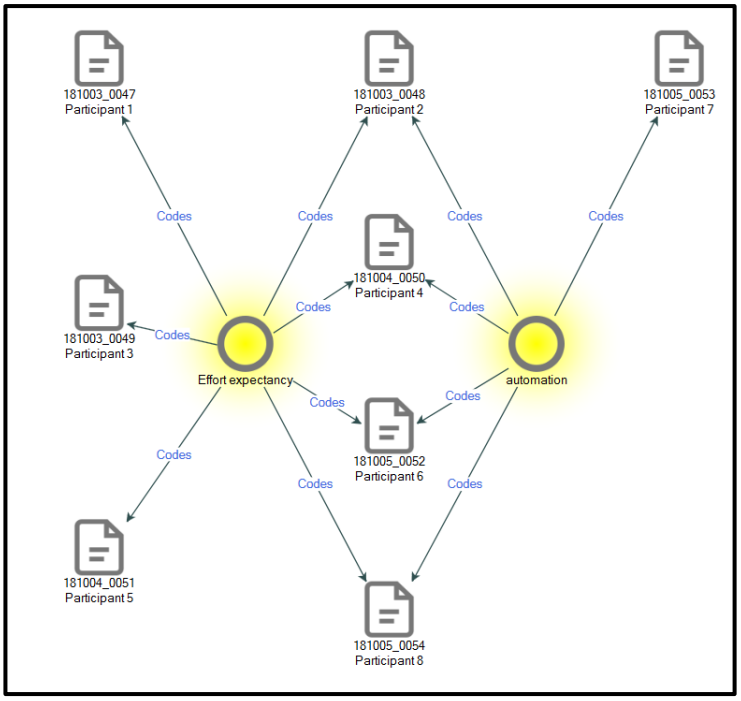

Figure 5.3: Joint References to Automation and Effort Expectancy

- Performance Expectancy vs Tools and technologies – as illustrated in Figure 5.4, 5 of the 8 participants made joint references to both these constructs in close proximity of each other:

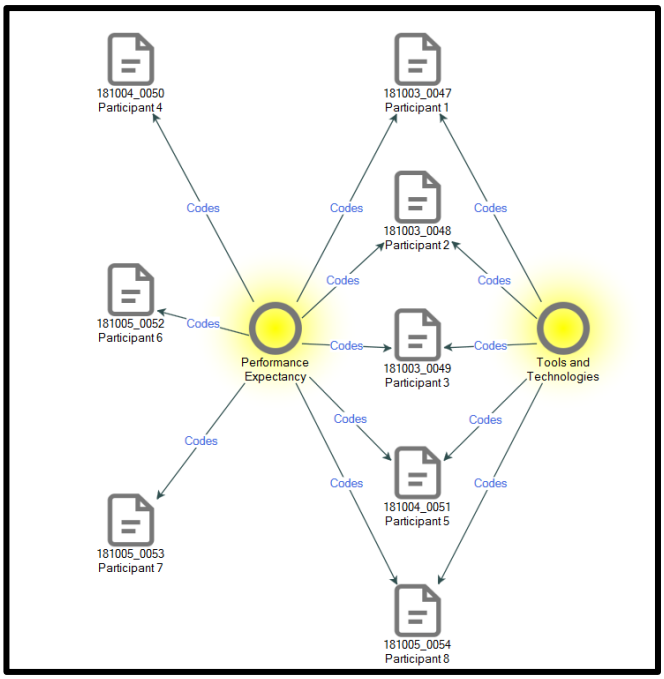

Figure 5.4: Joint References to Performance Expectancy vs Technologies

- Effort Expectancy vs Tools and technologies – as illustrated in Figure 5.5, 6 of the 8 participants made joint references to both these constructs in close proximity of each other:

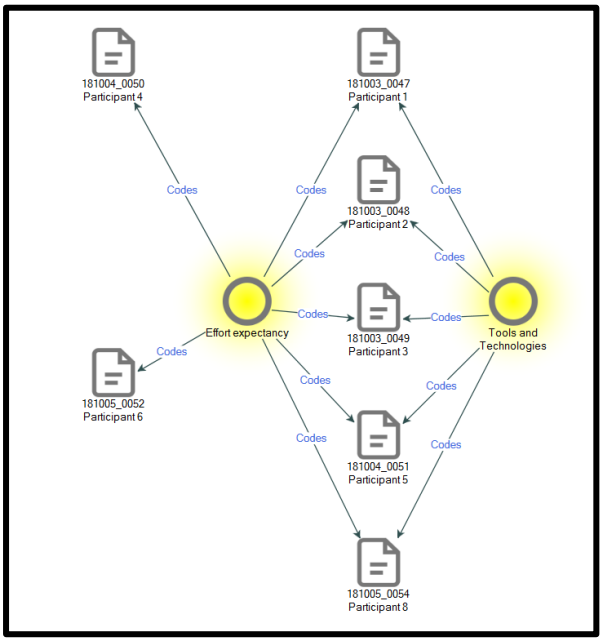

Figure 5.5: Joint References to Effort Expectancy vs Technologies

5.4 MIND MAP PRESENTATION

As discussed in Section 5.2, the graphical illustration of the main themes emanating from the qualitative analysis is presented in Figure 5.6 and Figure 5.7 as a mind map visualisation. Figure 5.6 has been structurally designed around the main constructs of DevOps ***definition***, ***strengths*** of using DevOps and ***challenges*** to DevOps implementation. Figure 5.6 represents a visual depiction of the textual narration that has dominated the discourse presented in the current chapter.

DevOps ***acceptance*** and ***success*** factors have been visually presented in Figure 5.7. As can be observed in Figure 5.7, the 6 acceptance factors are presented with the main themes that related to each factor. The preceding strategy is repeated for the 7 factors linked to success. Also, the 2 newly identified success factors of cloud and security are shown in Figure 5.7 (presented in landscape view for ease of viewing).

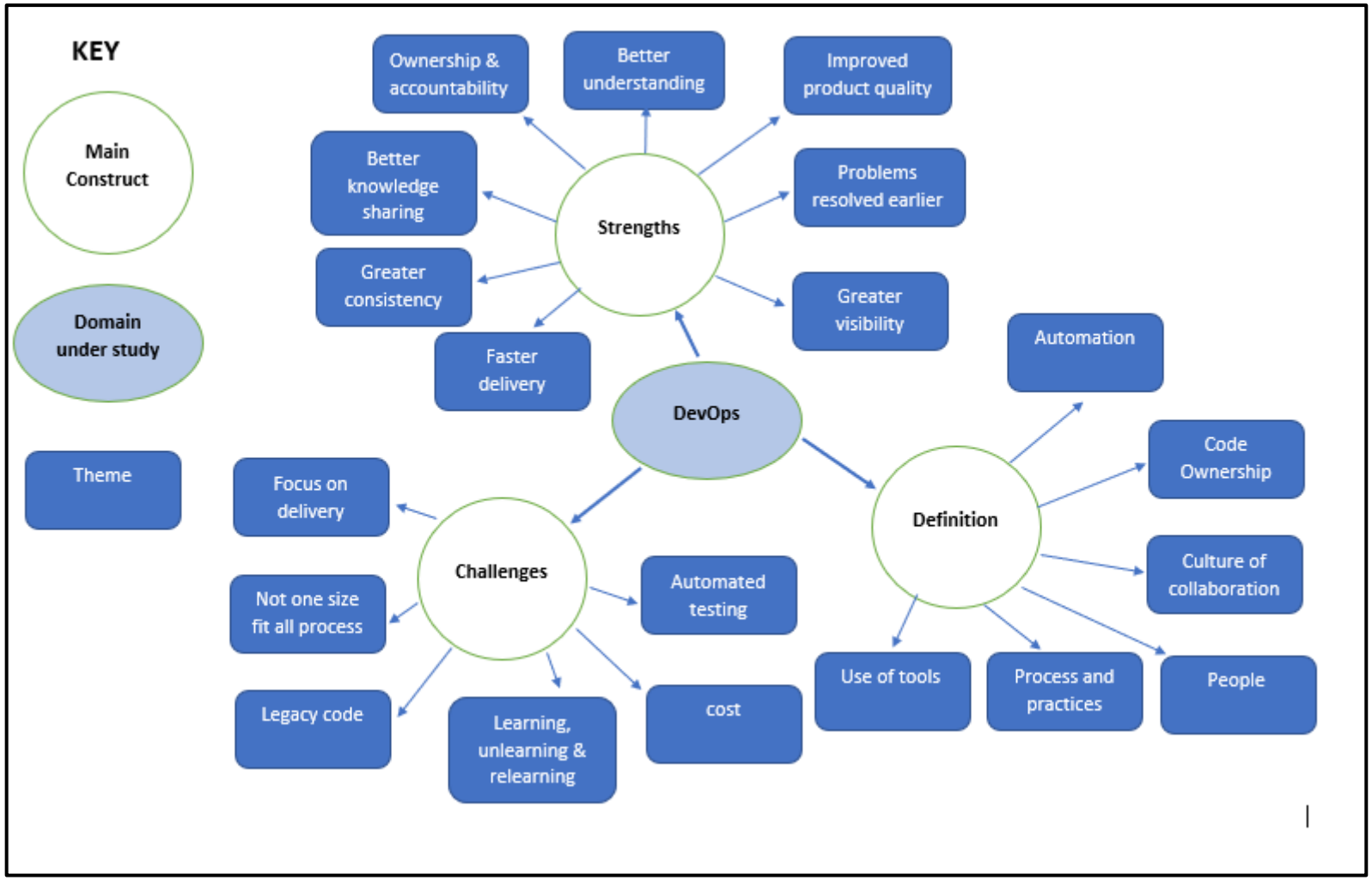

Figure 5.6: Thematic Mind Map of the Study's Main Constructs

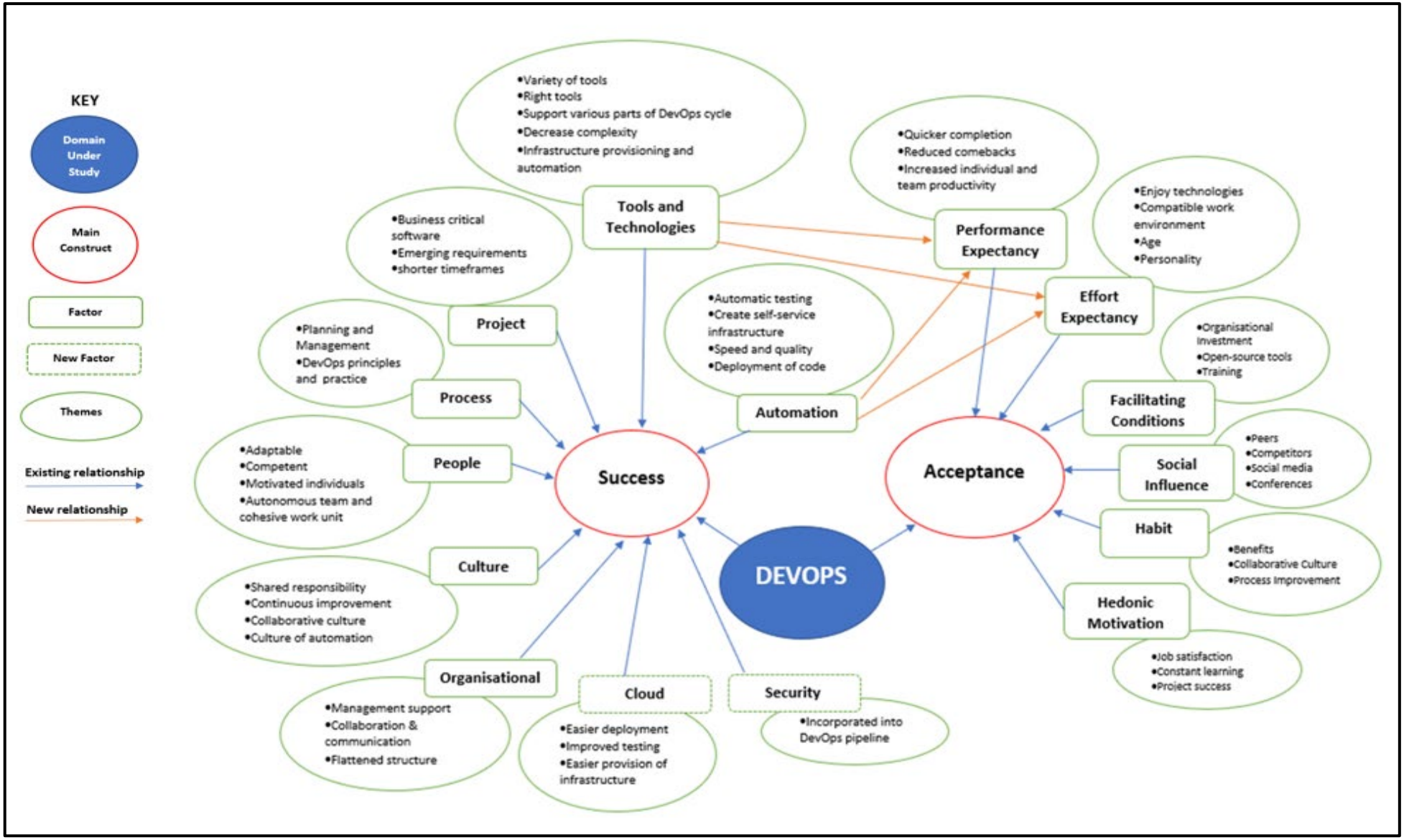

Figure 5.7: Thematic Mind Map of Success and Acceptance Factors

5.5 SUMMARY

This chapter used thick text descriptions to depict participants' lived experiences with DevOps. It highlighted participants' views on the definition of DevOps, its benefits, and the challenges to its adoption. Furthermore, important factors that highlight DevOp's success and acceptance were discussed. Using thick text descriptions enabled the researcher to portray the participants' experience, knowledge, and passion for DevOps. It also increases the external validity of the study in terms of transferability. The next chapter will use the results from this chapter to revise the conceptual model that will subsequently be validated quantitatively in later chapters.

CHAPTER 6: REVISED CONCEPTUAL MODEL AND STUDY'S HYPOTHESES

6.1 INTRODUCTION

The previous chapter analysed participants' qualitative responses concerning the definition of DevOps, its benefits, challenges in adoption, and the DevOps success and acceptance factors. This analysis confirmed factors from the preliminary conceptual model, derived new factors and established links between certain factors. This chapter discusses the themes for the success and acceptance factors that emerged from the qualitative analysis with supporting research studies to formulate the hypotheses, resulting in a revised conceptual model.

6.2 SUCCESS FACTORS

6.2.1 Success as a Dependent Variable

Chapter 3 highlighted various criteria used to measure success in software development. This section builds on these criteria within the context of DevOps. Senapathi et al. (2018) claimed that deployment frequency is a key success factor in software development. This also accords with Sallin et al. 's (2021) study, which identified deployment frequency and change failure rate as important criteria for success. Together with deployment frequency, Mishra and Otaiwi (2020) identified customer satisfaction, deployment success rates, and software quality as measures of DevOps success. In a related study, Batra and Jatain (2020) compared DevOps and the waterfall approach and found that DevOps improved software quality and increased customer satisfaction. Hence, deployment frequency, speed, software quality, customer satisfaction, and low change failure rate are important success measures.

6.2.2 Organisational Factors

Organisational factors refer to all the factors that affect the organisation's behaviour and everyone within it (Tsai, 2011). Hence, organisational factors can be conceptualised as operational characteristics, processes or circumstances as affirmed by Valaitis et al. (2018). Within the context of software development, organisational factors are the attributes, processes, and conditions that create an environment for the successful development of software. Factors such as executive support, cooperative culture, face-to-face communication, team co-location, committed sponsors, and reward

systems are critical organisational factors for software development success within organisations (Chow & Cao, 2008; Ramadan & Rizk, 2015).

Livermore (2007) provided evidence of a positive correlation between management support and Agile project success. Several studies (e.g. (Ambler, 2012; Hamdani & Butt, 2017; Mishra et al., 2021; Tanner & Seymour, 2014)) have affirmed that management support is essential for Agile success.

An organisational culture based on cooperation and collaboration rather than hierarchy is important to attaining success in an Agile project (Tanner & Seymour, 2014). Furthermore, Gandomani et al. (2014) and Koch (2005) state that this culture can only exist in teams if the cooperative and collaborative philosophy is embraced at an organisational level.

Face-to-face communication within project teams results in increased collaboration and speedier problem resolution, leading to the success of Agile projects (Hummel et al., 2015; Koch, 2005).

When changes are implemented in an organisation, reward systems can be introduced to motivate individuals and teams to adopt these changes and perform better. Research indicated that agile teams can become more productive when incentivised, resulting in agile success (Chow & Cao, 2008; Gandomani et al., 2014; Koch, 2005).

According to Chow and Cao (2008) and Schatz and Abdelshafi (2005), a committed sponsor is necessary for agile success. A committed sponsor is "someone who's willing to put everything on the line and is committed to moving to agile. Someone has to stand up to the critics, encourage the leaders, and communicate the team's vision" (Schatz & Abdelshafi, 2005, p. 38).

Co-location is also very important in ASD. Co-location enables teams to respond promptly to dynamic or uncertain requirements (Green et al., 2010).

In Agile software development, Misra et al. (2009) and Vithana et al. (2015) indicate that organisational factors are important for Agile success. Within DevOps, Van Belzen et al. (2019) and Azad (2022) indicate that organisational factors are essential for DevOps success. Additional qualitative findings from the current study found that management support, face-to-face interactions and cooperative culture are important factors for DevOps' success. These observations serve as a catalyst to generate a set of hypotheses pertaining to the influence of organisational factors on DevOps success. The null ($H1_0$) and alternate hypotheses ($H1_a$) are stated as:

$H1_0$: *Organisational factors do not influence DevOps success in the South African software industry*

$H1_a$: *Organisational factors positively influence DevOps success in the South African software industry.*

6.2.3 Culture Factors

According to Sharma and Coyne (2015), DevOps culture entails high collaboration across different roles in a software development environment. The most efficient processes or automated tools can be implemented within an organisation, but if the right culture does not exist, these tools and processes will be useless. Thus, having a DevOps culture is central to the adoption of DevOps. DevOps culture enables teams to be more efficient and effective in meeting customers' needs, increasing the quality of the software features developed, and meeting business objectives swiftly (Jha et al., 2023).

Luz and Pinto (2019) mention that a collaborative culture is core to adopting a DevOps strategy by breaking down the "wall" between development and operations teams. This will enable them to function as a DevOps team with a collaborative culture where all team members are on the "same page" and work in support of the same outcome. This will enable the delivery of stable, high-quality software and systems. When software fails or is not implemented successfully, developers, quality assurance, and operations commonly blame each other (Wahaballa et al., 2015). In a DevOps team, a shared understanding between key role players ensures a sharing of responsibility for the software they build. This means that the entire team takes responsibility for the success and failure of the developed software (Wiedemann et al., 2020).

Learning and continuous improvement are significant to the cultural aspect of DevOps. A DevOps organisation values and encourages learning to optimise performance, cost and speed of delivery (Katal et al., 2019). This results in continuous improvements within the DevOps process, resulting in significantly shorter update cycles (Ogala, 2022). DevOps teams are motivated to take reasonable risks and are not reprimanded when these risks do not result in a successful outcome (Savor et al., 2016).

Hence, DevOps teams should learn from past failures and successes to continuously improve. The deployment pipeline must be configured with fast feedback loops, allowing the DevOps philosophy of fail fast, recover fast to be practised (Mudadi & Lotriet, 2023; Soni, 2015). Another fundamental principle of the DevOps approach is automation that averts repetitive manual processes within the DevOps life cycle. As a result, automation will allow every step in the continuous integration and delivery pipeline to be repeated multiple times daily. This results in an increased rate of development, the generation of excellent quality software and an increased number of deployments (Arulkumar & Lathamanju, 2019). Hence, DevOps teams have a culture of automating almost everything to deploy software faster, safer and more reliably than ever.

The concept of culture or behavioural traits is important to DevOps success as alluded to by Erich et al. (2014); Lwakatare et al. (2016); Perera et al. (2016) and Rowse and Cohen (2021). Qualitative findings from this study also indicate that culture is essential for DevOps' success, with participants indicating that a collaborative working culture, shared responsibility, and continuous improvements are important for DevOps' success. These observations serve as a catalyst to generate a set of hypotheses pertaining to the influence of organisational culture on DevOps success. The null ($H2_0$) and alternate hypotheses ($H2_a$) are stated as:

$H2_0$: *Culture factors do not influence DevOps success in the South African software industry.*

$H2_a$: *Culture factors positively influence DevOps success in the South African software industry.*

6.2.4 People Factors

People play a critical role in the success of information system development; therefore, people must possess specific characteristics and abilities (Boehm & Turner, 2003). According to Cockburn, team members must have the necessary skills to achieve more value in less time (Cockburn & Highsmith, 2001). Cockburn and Highsmith (2001) and Lindvall et al. (2002) mentioned that software professionals must be competent with relevant work experience in the technology domain. Due to the constant technological changes, software developers must be willing to learn and innovate continually (Lindvall et al., 2002; Misra et al., 2009).

When working in a software development environment, the team and its members must have self-organising skills so they are clear on what needs to be achieved and how they are going to achieve it (Cockburn & Highsmith, 2001; Reifer et al., 2003). To enable this, many managers moved from micromanaging project teams to giving individual team members more autonomy in deciding what and how tasks will be completed (Augustine et al., 2005).

Management personnel must have the skills to manage projects with a light -touch. If they control projects heavy-handedly with too much structure, it will compromise flexibility and creativity in the team (Nayak & Patra, 2001). This strategy of upholding flexibility and agility is one that can be traced to the contributions made by the pioneers of Agile methodology. Ambler (2005), as well as Boehm and Turner (2003), were strong advocates of the theory that agile will succeed if the participants are interactive, and possess good collaboration and communication skills. Kent et al. (2001) indicated that individual motivation is essential in project success since it promotes collaboration, responsibility and encourages the willingness to learn from each other. These dated references are corroborated in later studies that may be deemed to be more current (e.g. Abd et al. (2016); Aldahmash et al. (2017); Bogopa and Marnewick (2022); Chiyangwa (2017); Muhammad et al.

(2021); Ramadan and Rizk (2015); Vithana et al. (2015)) where it was established that collaboration and synergy between people in an Agile and DevOps environment is important for Agile success.

Participants from the qualitative phase of the current study also indicated that DevOps professionals should be adaptable and learn to adjust to different circumstances, work with others as a cohesive team, and be competent and motivated. These observations served as a catalyst to generate a set of hypotheses pertaining to the influence of people on DevOps success. The null ($H3_0$) and alternate hypotheses ($H3_a$) are stated as:

$H3_0$: *People factors do not influence DevOps success in the South African software industry.*

$H3_a$: *People factors positively influence DevOps success in the South African software industry.*

6.2.5 Process

A software process consists of various methods, activities and techniques used during software development (Pressman, 2010). In an Agile software development environment, processes that manage the overall project, requirements and configuration changes are central to project success (Chow & Cao, 2008; Koch, 2005). According to Koch (2005), project management will allow for proper scope definition, resource allocation, risk management and communication, while requirement management will ensure proper elicitation, validation, change control and traceability of requirements. Furthermore, Koch (2005) mentioned that configuration management promotes appropriate version control, change management, and software process auditing.

Apart from management processes, modern-day software development must incorporate processes to communicate effectively within the project team, the development team, and the customer. This is achieved in Agile methodologies such as Scrum by engaging in process-driven behaviour such as daily standup meetings and sprint cycles and sprint reviews. Increased communication allows for rapid resolution of problems and increases the likelihood of project success and customer commitment, and it presents a mechanism to ensure that team members honour a regular working schedule. The relevance of these process-oriented activities is confirmed in studies by Bogopa and Marnewick (2022) and Muhammad et al. (2021) where it was found that process factors are important to software development process success.

The DevOps lifecycle incorporates processes such as release management, continuous integration, continuous delivery, continuous testing and monitoring. Release management is preparing, coordinating, and releasing software to production. Assuring the dependable, efficient, and timely delivery of new features, bug fixes, and enhancements to end-users is the primary focus of release management. DevOps' release management is essential because it ensures that software is provided to end-users in a timely and dependable way, with as few bugs and outages as possible. The DevOps

initiatives are managed and planned with end-to-end traceability across all processes, driving release roadmaps, project plans, and delivery timelines (Lakhani, 2023; Mohammad, 2019; Sharma & Coyne, 2015; Watts, 2023).

Continuous integration is a process of developer's code being continuously integrated and validated with each other's. This results in the early identification of issues, thereby reducing risk in the DevOps lifecycle (Almeida et al., 2022; Stahl et al., 2017). Continuous integration enables continuous delivery that results in automated software deployment to the unit testing, integration testing, system testing, staging and production environments (Mowad et al., 2022). This increases efficiency, leading to faster delivery of software. Therefore, DevOps projects follow the practice of continuous delivery thereby ensuring that the software can be released at any time (Sharma & Coyne, 2015).

Embedded within continuous integration and continuous delivery are processes for continuous testing. These include the provisioning and configuration of the test environment, management of test data and the different types of testing such as integration functional, performance and security (Sachdeva, 2016). The DevOps projects follow a practice of continuous and automated testing. This allows for testing early, testing often and testing everywhere, thereby enabling continuous quality and improvement (Sharma & Coyne, 2015)

Continuous monitoring processes are also incorporated within DevOps. This allows every activity to be closely monitored, enabling organisations to provide high-quality software with little disruptions. It also shortens the time it takes to find and fix issues in production and adopts proactive efforts to address operational and quality flaws throughout the DevOps lifecycle. This method results in more dependable and efficient software delivery (Bucena & Kirikova, 2017; López-Peña et al., 2020; Parashar, 2022)

The pivotal role played by DevOps processes (as identified in the literature) is corroborated in the current study's qualitative findings that are suggestive of the theory that DevOps processes (that embed DevOps principles and practices) are essential for DevOps' success. These observations serve as a catalyst to generate a set of hypotheses pertaining to the influence of DevOps processes on DevOps success. The null ($H4_0$) and alternate hypotheses ($H4_a$) are stated as:

$H4_0$: *DevOps processes have no influence on DevOps success in the South African software industry.*

$H4_a$: *DevOps processes have a positive influence on DevOps success in the South African software industry.*

6.2.6 Project Factors

The characteristics of the project can influence the success of the software development effort. A prescriptive process model will be more suitable than an agile process for mission-critical and documentation-intensive projects, and all requirements need to be known upfront (Cohen et al., 2004). Agile process is preferred for unclear or changing projects due to market changes. Hence, the software process must be selected based on the project characteristics to succeed. A systematic review by Abd et al. (2016) identified project type, schedule and team size as essential items within the project factor. In an agile environment, the agile project type must be non-life-critical (Chow & Cao, 2008; Cohen et al., 2004; Koch, 2005). Cohen et al. (2004) stated that agile is unsuitable for projects with criticality, reliability, and safety requirements.

Furthermore, the Agile project must vary in scope and requirements, evolve during project development and be suitable for dynamic environments. Agile is best suited for projects with an accelerated schedule so that software can be given to clients early, allowing them to achieve immediate benefit (Chow & Cao, 2008). Small teams aligned to small projects are appropriate for Agile projects. These small teams have less than 20 members, and the team members need to have all the necessary skills and knowledge to work as one cohesive unit. Besides project type, schedule and team size, researchers suggest that cost evaluation and risk analysis are important for project success.

Like Agile projects, DevOps projects have a dynamic, accelerated schedule, empowering organisations to promptly address market demands and optimise software delivery processes (Hurkens, 2019). DevOps projects also enable changing system requirements and varying scope by promoting a culture of cooperation, ongoing feedback, and adaptability. This capability allows teams to adapt to changes and effectively provide value to their stakeholders (Angara et al., 2020; Angara, 2020). A DevOps project mandates risk assessment in the beginning (during continuous integration) and continuously promotes the practice of identifying and managing risks throughout the entire development lifecycle (Bigham & Litchfield, 2022; de Roos, 2020). DevOps projects are highly compatible with mission-critical software for businesses because they emphasise agility, collaboration, continuous monitoring, and speed. This assists organisations in delivering, maintaining, and improving software applications critical to their businesses' success and functioning (Chawre, 2023; Gall & Pigni, 2018). Their literary contributions are corroborated empirically by the current study's qualitative data, which established that project types such as business-critical software projects with short time frames and projects with emerging requirements are important factors that influence DevOps' success. These observations serve as a catalyst for generating a set of hypotheses

pertaining to the context and attributes of a project and DevOps success. The null ($H5_0$) and alternate hypotheses ($H5_a$) are stated as:

$H5_0$: *Project factors have no influence on DevOps success in the South African software industry.*

$H5_a$: *Project factors positively influence DevOps success in the South African software industry.*

6.2.7 Tools and Technologies

Technologies play a crucial part in the DevOps architecture. The initial stage in the DevOps lifecycle is planning, which entails defining the project's scope, needs, resources, and schedule. The primary emphasis is establishing and adjusting corporate objectives based on client input (Pennington, 2019; Sen et al., 2021). This phase utilises various tools such as Jira and confluence (Flefil et al., 2022). Confluence and Jira revolutionise team collaboration, software development, and decision tracking (Senapathi et al., 2018). Jira is highly effective in assisting DevOps teams with the planning and monitoring project tasks (Ortu et al., 2016). Confluence is a collaborative application that uses wikis to facilitate teamwork and enable efficient organisation of ideas, material, and files (Srivastava, 2023).

In the development phase, the requirements collected during the requirements phase are transformed into code, constructed, tested, and made ready for deployment. During this phase, CI/CD pipelines are established to automate the activities of coding, building, testing, and deployment. These pipelines are established through the utilisation of diverse techniques and technologies. Throughout the construction process, version control technologies like Git oversee the administration of the source code or scripts (Donca et al., 2022). Multiple developers can collaborate on a shared project and monitor modifications to the codebase using these software utilities. In addition, developers can write and test functionalities on separate branches and then integrate these branches into the main branch (Govil et al., 2020; Perveez, 2023; Shah et al., 2018). By implementing CI/CD, DevOps can provide code quickly and efficiently. Jenkins is a widely used tool for developing and orchestrating a continuous integration and continuous delivery (CI/CD) pipeline (Mysari & Bejgam, 2020). In the CI/CD process, when code is committed to Git, Jenkins initiates building and testing the code.

Upon completing the build process, Jenkins will deploy the source code to the test server and inform the deployment team. In case of a build failure, Jenkins will inform the developer team about the issues (Hamilton, 2023). Maven and Gradle are automation tools that streamline and supervise the build process. This is accomplished by optimising the process of managing project dependencies. The software automatically retrieves the necessary libraries from remote repositories, removing the need for developers to download them from other sources manually. In addition, they possess a build lifecycle consisting of the following stages: compilation, testing, packaging, and deployment, which

are conducted automatically. This aids developers by avoiding the need to write redundant build scripts (McKenna, 2022).

JUnit and Selenium play significant roles in the pursuit of automating testing procedures and guaranteeing software quality across various tiers of the application architecture. JUnit primarily emphasises testing individual units and components (Rodrigues, 2016), whereas Selenium is designed explicitly to test user interfaces and end-to-end functionalities (Agarwal et al., 2018). These technologies contribute to increased quality assurance and dependability of software being disseminated inside the DevOps pipeline (Verona, 2016).

Operational tools such as Chef are necessary in DevOps since they define and manage infrastructure using Infrastructure as Code (IaC), allowing for consistency and repeatability in the infrastructure setup. It also manages the correct and consistent configuration of servers, reducing the risk of configuration changes, thereby making it easier to recover from failures. It allows for automatic provisioning of servers, software installation, and configuration updates. Operational tools like Chef also integrate well with the CI/CD by providing consistent environments across development, testing, and production stages (Katal et al., 2019; Srivastava et al., 2023).

Deployment tools such as Docker use containerisation to package applications for deployments. Containerisation creates standalone containers to encapsulate everything needed to run an application, including libraries, configurations, and runtime environments. This prevents the problem of a deployment running on one machine and not another. Within the CI/CD pipeline, deployment tools ensure that the application behaves consistently across different stages in the pipeline (Hardikar et al., 2021; Narasimhulu et al., 2023).

Container orchestration plays a vital role in DevOps by facilitating the efficient administration and automation of applications encapsulated within containers' deployment, scalability, and administration. Kubernetes, an open-source container orchestration platform, is extensively utilised in DevOps. Kubernetes facilitates the management of containers, encompassing many functionalities such as deployment, scalability, rolling updates, and load balancing. This aspect holds significant importance in a DevOps setting characterised by frequent application modifications and updates. Kubernetes possesses the capability to autonomously scale infrastructure by considering resource use or established rules, hence guaranteeing optimal performance without the need for manual intervention (Hoque et al., 2017; Shah & Dubaria, 2019).

Monitoring technologies such as Splunk play a crucial part in DevOps initiatives by offering immediate visibility into failures and problems throughout the infrastructure, hence contributing to the overall success of the projects. These tools facilitate DevOps teams' rapid identification and

resolution of issues, enabling them to effectively address high-impact problems and maintain the dependability and efficiency of their applications and systems. (Ebert et al., 2016). Monitoring technologies are designed to consistently gather and evaluate data from many sources encompassing the infrastructure, applications, and services. The utilisation of real-time data facilitates the ability of DevOps teams to identify and address issues as they arise promptly, hence reducing the response time and mitigating instances of system unavailability (Akshaya et al., 2015; Katal et al., 2019; Zamfir et al., 2019). Monitoring tools are designed in such a way that they can generate alerts and notifications if predetermined thresholds or circumstances are satisfied. Implementing such a proactive process enables DevOps teams to proactively identify and resolve possible issues before negatively impacting consumers (Yarlagadda, 2021). Splunk is a software solution that facilitates the consolidation of log data from several origins, enhancing the efficiency of searching, analysing, and correlating logs derived from different elements within the application stack. This process facilitates identifying and resolving issues and examining underlying causes (Jha et al., 2023).

Azad (2022) indicated that technology is one of the essential factors in DevOps' success. Qualitative findings from the current study indicate that technologies in DevOps decrease complexity in the development process by decreasing manual tasks and providing the necessary support to enable infrastructure provisioning and automation. Furthermore, technologies provide support for the entire DevOps lifecycle. These observations serve as a catalyst for the generation of a set of hypotheses pertaining to the role that technologies play in the success of DevOps projects. The null ($H6_0$) and alternate hypotheses ($H6_a$) are stated as:

$H6_0$: *The use of technologies has no influence on DevOps success in the South African software industry*

$H6_a$: *The use of technologies has a positive influence on DevOps success in the South African software industry*

6.2.8 Automation

Automation enhances software quality within a DevOps environment (Waller et al., 2015). In order to establish an automated framework for DevOps, it is imperative to have the appropriate individuals, procedures, and technologies in place (Shah et al., 2018). Enabling automated processes across all settings—from continuous integration and testing to real-world production deployments makes it possible to improve and monitor performance across the entire DevOps lifecycle (Perera et al., 2017; Waller et al., 2015).

Triggers are critical to DevOps automation since they initiate automated processes (Salameh, 2019). The primary purpose of these triggers is to streamline and accelerate the delivery procedure, enhance uniformity, and diminish the necessity for human intervention (Faustino et al., 2022).

During continuous integration, a code commit to a version control system initiates automated procedures, including code compilation, testing, and packaging. Furthermore, automatic notifications are sent to developers when the build is successful or unsuccessful (Fitzgerald & Stol, 2017; Shahin et al., 2017; Soni, 2015).

Automated continuous testing starts with several triggers across the DevOps lifecycle (Mascheroni & Irrazábal, 2018). These triggers ensure that testing is essential to the development and helps identify issues early. Automation checks new codes for regression faults after committing a code (Verona, 2016). The CI build also runs unit, integration, and interface tests. Automated tests can evaluate application functionality and performance after a successful build and staging deployment during continuous delivery (Ramler et al., 2014). Continuous monitoring also uses automation to activate triggers at predetermined thresholds (Giamattei et al., 2024).

UI testing is a process that evaluates the user interface of an application. Automated UI testing can be used to establish test environments that automatically replicate user interactions. This facilitates comprehensive application user interface testing, ensuring end-to-end evaluation (Millan, 2022).

Modification of the code repository can introduce new faults in software. Regression testing is used to detect if such errors are introduced. This is achieved by automating the configuration of the test suites to execute regression testing iteratively to detect and address any potential regression issues (Sachdeva, 2016).

Performance monitoring entails systematically observing and measuring an application's response times, resource use, and other performance parameters. Automated performance monitoring solutions are the ongoing process of collecting and analysing performance data from the application. Thresholds and baselines are implemented to activate notifications when performance metrics deviate from expected values. Notification alerts can take various forms, such as those delivered through email, SMS, or chat platforms (Alves, 2022; Perera et al., 2017).

There are various automated error tracking and logging processes in DevOps that are capable of capturing problems in real-time. DevOps teams establish mechanisms to promptly notify stakeholders of critical issues, enabling prompt investigation and resolution (Ahmed et al., 2021; Capizzi et al., 2020; Hrusto et al., 2023).

Infrastructure as code (IaC) refers to utilising code to automate the provisioning and management of computing infrastructure (database systems, servers, and networks) rather than relying on manual

processes and configurations. Infrastructure as Code (IaC) is used in DevOps to establish the infrastructure's intended configuration. IaC can be autonomously deployed, ensuring consistent and replicable environments (Artac et al., 2017; Siebra et al., 2018). The configuration of CI/CD pipelines automates the deployment of application code to automatically provisioned server instances. This guarantees that every modification to the code can be implemented in production promptly and consistently (Donca et al., 2022). By dynamically scaling infrastructure resources to accommodate fluctuations in application demand, automated server provisioning ensures optimal resource utilisation (Bahadori & Vardanega, 2019; Thammareddi et al., 2023). Applications and their dependencies are also encapsulated by containerisation to facilitate easy deployment. This functionality allows containerised applications to be deployed automatically across various server environments (Narasimhulu et al., 2023).

Perera et al. (2016); Gupta et al. (2017) and Mishra and Otaiwi (2020)mentioned the importance of automation in DevOps. Participants in this study explained that automation enables automatic deployment of code, self-service infrastructure provisioning, automated testing and support for the entire DevOps pipeline. These responses served as a catalyst for the generation of a set of hypotheses pertaining to the role that automation plays in the success of DevOps projects. The null ($H7_0$) and alternate hypotheses ($H7_a$) are stated as:

$H7_0$: *Automation does not have an influence on the success of DevOps projects in the South African software industry.*

$H7_a$: *Automation has a positive influence on the success of DevOps projects in the South African software industry.*

6.2.9 Cloud Computing

Cloud platforms are critical in promoting the advancement of automation methodologies across DevOps's entire development and operations lifecycle (Battina, 2020). Cloud-based continuous integration and continuous deployment (CI/CD) services like AWS Code Pipeline, Azure DevOps, and Google Cloud Build are critical in automating DevOps's entire software delivery pipeline. These services enable software's efficient and effective release by facilitating the seamless transfer of code commits to production deployments (Buttar et al., 2023; Priyadarsini et al., 2020). Infrastructure as Code (IaC) services such as AWS CloudFormation, Azure Resource Manager, and Google Cloud Deployment Manager are available on cloud platforms such as AWS, Azure, and Google Cloud. These services enable the automatic generation and administration of infrastructure resources using code, ensuring consistency and reproducibility. Cloud systems support automatic scaling, allowing

applications to alter resource allocation in response to variable workloads and improve operational efficiency (Achar, 2021).

Container orchestration solutions, such as Kubernetes, are often hosted on cloud infrastructures to assist in the automation of different activities, such as deploying, scaling, and managing containerised applications (Miller et al., 2021; Rodriguez & Buyya, 2020).

Cloud-managed database services like Amazon RDS, Azure Database, and Google Cloud SQL automate database processes, including provisioning, backups, scaling, and patching (Oles, 2018; Sreyes et al., 2022).

Cloud-native monitoring and analytics services like AWS CloudWatch, Azure Monitor, and Google Cloud Monitoring provide automatic insights into the health of applications and infrastructure, allowing for proactive problem discovery and remediation (Ågerfalk et al., 2009; Chauhan & Shiaeles, 2023; Sreyes et al., 2022).

Cloud-based testing environments, such as AWS Test Environments and Azure DevTest Labs, are used by practitioners to distribute and supervise testing infrastructure efficiently as needed. Incorporating automated testing tools into these systems makes test execution more efficient. DevOps teams use cloud infrastructure to allocate and set up many test environments efficiently. This enables DevOps teams to run tests concurrently, reducing testing time and a faster feedback process. It is usual practice to guarantee that one cloud-based test environment is adequately isolated from others, allowing autonomous testing to take place without intervention or disturbance. The state of isolation improves the dependability of test results. It is also typical to associate each phase of a Continuous Integration/Continuous Deployment (CI/CD) pipeline with distinct cloud-hosted test environments. This method thoroughly tests the code before moving on to the next level.

Cloud provides various security services (for example, AWS Identity and Access Management, Azure Security Center, and Google Cloud Identity) to improve the security posture of production workloads through access control, threat detection, and compliance enforcement (Chauhan & Shiaeles, 2023).

According to participants in this study, the cloud provides better automation abilities, a more efficient and effective testing environment, and the capability of rapidly creating consistent infrastructure. These responses pertaining to the automation potential intrinsic to cloud computing served as a catalyst for the generation of a set of hypotheses that allude to the influence of cloud computing on DevOps success. The null ($H8_0$) and alternate hypotheses ($H6_a$) are stated as:

$H8_0$: *Cloud computing does not have an influence on the success of DevOps projects in the South African software industry.*

$H8_a$: *Cloud computing has a positive influence on the success of DevOps projects in the South African software industry.*

6.2.10 Security

Code analysis is a fundamental approach employed in DevOps projects to conduct vulnerability testing and ensure quality assurance. Automated code analysis tools are crucial in identifying security vulnerabilities and code quality concerns and evaluating adherence to coding standards. Automated code analysis techniques, specifically static application security testing (SAST) tools, are employed to thoroughly examine the source code to detect and pinpoint potential security flaws (Rajapakse et al., 2021). These software tools assess the codebase for possible concerns such as SQL injection, cross-site scripting (XSS), and other prevalent security flaws (Algaith et al., 2018). Code analysis tools are essential in ensuring code quality and compliance with coding standards. Static code analysis is conducted in order to detect coding trends that have the potential to result in errors or present issues in terms of maintenance (Charan et al., 2023; Moyón et al., 2020). Change management techniques are implemented in DevOps initiatives to enhance the effective management of security modifications that are crucial for the project's success. These protocols facilitate the capacity of developers to propose security upgrades or significant changes following established best practices while ensuring the maintenance of operational stability (Zaydi & Nassereddine, 2020). This procedure guarantees that security improvements undergo comprehensive evaluation, experimentation, and implementation while upholding operational stability and adherence to established standards. It also incorporates the principle of "shifting-left", whereby security is addressed early in the software development cycle (Percival, 2020; Vardhan, 2023). In the context of DevOps initiatives, it is of utmost importance to consistently evaluate newly introduced features and updates to mitigate emerging dangers effectively, ensuring the program's ongoing security and robustness. This methodology encompasses threat modelling, vulnerability assessments, and security testing (Zaydi & Nassereddine, 2020). The primary objective of these exercises is to evaluate and analyse the potential security ramifications that may arise from introducing new features or implementing updates. This facilitates the early detection of potential dangers throughout the development phase (Paule et al., 2019). Automated vulnerability assessment solutions are seamlessly included in the DevOps pipeline to conduct comprehensive scans of code, dependencies, and configurations within the DevOps framework. These tools are designed to detect and analyse system vulnerabilities and their potential impact on the system (Maayan, 2023).

Security is incorporated into each development lifecycle stage, including testing, deployment, planning, and coding. This is accomplished through shift-left security, which entails implementing security controls and practices during the earliest stages of development. This is achieved by

integrating security controls and assessments during the design and coding phases (Lombardi & Fanton, 2023; Yarlagadda, 2020). Security issues are detected and remedied during development with automated code analysis tools and training on secure coding practices for developers (Ashenden & Ollis, 2020). CI/CD pipelines incorporate automated security testing, consisting of interactive application security testing (IAST), dynamic analysis, and static analysis, to detect vulnerabilities early (Abiola & Olufemi, 2023).

Periodic scans, code reviews, and penetration testing are crucial in DevOps projects to maintain continuing security and quality, even after the code has been released. These techniques aid in identifying and resolving vulnerabilities and issues that may arise after the release. Automated vulnerability scanning technologies examine production systems and applications for flaws. The findings aid in detecting and resolving recently discovered weaknesses that may have arisen subsequent to the initial implementation (Kalaiselvi, 2020). Code review processes persist even after the code has been implemented. Modifications implemented in later revisions undergo scrutiny from fellow software developers and automated code analysis tools to guarantee the integrity and safety of the code. Periodic penetration tests are undertaken to assess the efficacy of security safeguards. This aids in the identification of vulnerabilities that arise as a result of modifications to the code or configuration changes (AWS, 2023). Security embedded in the entire DevOps pipeline is essential for DevOps success based on responses from participants in this study. The preceding claim leads to a set of hypotheses where the null ($H9_0$) and alternate hypotheses ($H9_a$) are stated as:

$H9_0$: *Security does not have an influence on the success of DevOps projects in the South African software industry.*

$H9_a$: *Security has a positive influence on the success of DevOps projects in the South African software industry.*

6.3 ACCEPTANCE FACTORS

6.3.1 Actual Usage

According to Al-Gahtani (2001) and Davis (1989), the acceptance of IS is measured by its' frequency of use. Kim and Gyo (2008) further explained that the actual usage behaviour of a technology can be effectively determined based on frequency of use. Therefore, the use of DevOps in software development, particularly the actual behaviour, performs as a dependent variable in this study and can be determined by use frequency.

6.3.2 Behavioural Intention

BI refers to an individual's willingness to adopt and use new technology. It is an important factor that determines the actual usage behaviour of technology. Ajzen (1991, p. 181) argues that "Intentions are assumed to capture the motivational factors that influence a behaviour. They are indications of how hard people are willing to try, or how much of an effort they are planning to exert in order to perform the behaviour". The BI construct in the UTUAT model is comparable to the attitude towards behaviour in the TRA, TPB and DTPB models. The Motivation Model (MM) also captures BI using the constructs of extrinsic and intrinsic motivation. Recent studies (e.g. (Al-Mamary, 2022; Alkhowaiter, 2022; Raman et al., 2022)) demonstrated BI to influence technology usage positively. Within the context of Agile software development, Mudarikwa and Grace (2018) studied the acceptance of Agile development in banking and found that BI had a positive influence on usage of Agile methodology. An extension of this outcome is to propose a similar hypothesis that relates to DevOps. The null ($H10_0$) and alternate hypotheses ($H10_a$) are stated as:

$H10_0$: *BI has no influence the actual usage of DevOps.*

$H10_a$: *BI will positively influence the actual usage of DevOps.*

Besides being a dependent factor determining actual usage, the BI factor is also an independent factor influenced by other factors. Each of these factors is discussed below.

6.3.3 Performance Expectancy

Venkatesh et al. (2003, p. 447) define performance expectancy (PE) as "the degree to which an individual believes that using a system will help him or her to attain gains in job performance". PE is based on constructs in previous models and theories, such as PU in TAM, extrinsic motivation in MM, relative advantage in DOI, job fit in MPCU and outcome expectancy in SCT (Abubakar & Ahmad, 2013). According to Davis (1985), PU refers to the extent to which people think the system can enhance their job performance or increase productivity. Extrinsic motivation is when users are motivated to use a new technology to gain external rewards, and it is known to increase performance and productivity (Ryan & Deci, 2000). Job fit is achieved when new technology is aligned with an individual's job, resulting in increased performance and productivity (Thompson et al., 1991). In SCT, outcome expectancy measures the user's expectation that using a new technology can result in a positive outcome, such as increased productivity and performance (Bandura, 1986a). Moore and Benbasat (1991) define relative advantage as the extent to which a new innovation is more beneficial than its predecessor. According to Venkatesh et al. (2012), research has demonstrated that PE positively influences BI and, therefore, strongly predicts BI. Recent studies by Alomari and Abdullah (2023); Ariza-Montes et al. (2023) and Chatterjee (2022) provide further support that PE can

positively influence BI. Within the domain of software development studies by Karunarathna et al. (2023); Mamakou (2023) and Huq (2017) demonstrated the positive relationship between PE and BI. Venkatesh et al. (2016) also indicate that PE significantly predicts BI in mandatory and non-mandatory settings. In this study, PE is defined as the degree to which software professionals perceive that using DevOps will result in positive project outcomes and increased job performance. Qualitative responses by professionals in this study also indicated that DevOps proved to be quite effective by ensuring quicker project completion and increased individual and team productivity. It is this kind of outcome served as a catalyst for the generation of a set of hypotheses where the null ($H11_0$) and alternate hypotheses ($H11_a$) are stated as:

$H11_0$: *PE does not influence the BI of software professionals to use DevOps.*

$H11_a$: *PE positively influences the BI of software professionals to use DevOps.*

6.3.4 Effort Expectancy

Effort expectancy (EE) is defined as "the degree of ease associated with the use of the system" (Venkatesh et al., 2003, p. 450). EE is rooted in the PEOU construct in TAM and complexity in MPCU and DOI. PEOU measures how easy it is to use a new technology (Davis, 1989). Similarly, complexity determines users' perception of how easy or difficult it is to use an innovation (Rogers, 1967; Thompson et al., 1991). Venkatesh et al. (2012) asserts that empirical evidence supports the notion that EE positively impacts BI. Recent studies by Shane et al. (2022) and Jameel et al. (2022) further demonstrated EE influence on BI. This, therefore, makes EE a reliable predictor of BI. Studies by Chiyangwa (2017) and Huq (2017) demonstrated the positive relationship between EE and BI within the domain of IS development. In this study, PE is defined as the degree to which software professionals perceive that using DevOps is easy. The qualitative analysis for EE in this study showed that DevOps professionals perceived DevOps to be easy to use, especially for professionals who enjoyed working with technologies or previously worked in an environment comparable to DevOps. The preceding outcome served as a catalyst for the generation of a set of hypotheses where the null ($H12_0$) and alternate hypotheses ($H12_a$) are stated as:

$H12_0$: *EE does not influence the BI of software professionals to use DevOps.*

$H12_a$: *EE positively influences the BI of software professionals to use DevOps.*

6.3.5 Facilitating Conditions

Facilitating conditions are users' perceptions regarding the availability of necessary infrastructure and support to adopt technology successfully (Venkatesh et al., 2003). According to Gupta and Dogra (2017), facilitating conditions are vital to the acceptance of new technology because they either

support or limit its adoption. Yeh and Tseng (2017) further assert that facilitating conditions should be a top priority, as the better the conditions are, the easier it will be for consumers to adopt the technology. Oliveira et al. (2016) concurred with this finding, adding that if the necessary infrastructure and support are in place to facilitate technology adoption, the tendency to adopt the technology tends to increase. The facilitating conditions construct is derived from the compatibility, perceived behavioural control, and facilitating conditions constructs found in TPB (Ajzen, 1991), CTAMTPB (Taylor & Todd, 1995b), MPCU (Thompson et al., 1991), and IDT (Rogers, 1983). Perceived behavioural control pertains to an individual's subjective evaluation of the ease or difficulty of executing a specific behaviour and is moderated by resource and technology facilitating conditions. Facilitating conditions are factors in the environment that boost people's willingness to utilise technology, such as prompting for help when the system is challenging. Compatibility evaluates the extent to which the innovation aligns with organisations' or individuals' existing practices and values. Hence, for the innovation to be successfully adopted, the environment must be conducive to the values and expectations of the adopter. According to Venkatesh et al. (2003), when the constructs performance expectancy and facilitating conditions are in the same model, facilitating conditions become insignificant in determining BI. However, facilitating conditions have a significant effect on usage. Recent studies by Kadim and Sunardi (2023) and Tseng et al. (2022) showed that facilitating conditions positively influence actual usage. Furthermore, Lambert (2011) and Mudarikwa and Grace (2018) found that facilitating conditions positively influence the BI to adopt Agile. These results served as a catalyst for the generation of a set of hypotheses where the null ($H13_0$) and alternate hypotheses ($H13_a$) are stated as:

$H13_0$: *Facilitating conditions do not influence the BI of software professionals to use DevOps.*

$H13_a$: *Facilitating conditions positively influences the BI of software professionals to use DevOps.*

6.3.6 Social Influence

Venkatesh et al. (2012, p. 159) defined SI as "the degree to which consumers perceive that important people in their lives (family and friends) believe that they should use a technology". Social influence is comparable to the subjective norms (TRA, TAM2, TPB, and DTPB), social factors (MPCU), and image (IDT) constructs in that they all indicate how individuals' conduct is influenced by how others perceive them (Venkatesh et al., 2003). Subjective norm refers to a person's perception that the majority of those who are significant to them believe that they should or should not engage with the technology (Taylor & Todd, 1995b; Venkatesh & Davis, 2000). Behaviour is influenced by social factors, indicating that it is determined by individuals' adherence to societal norms and beliefs, which are derived from the influence of others (Thompson et al., 1991). On the other hand, image is how

much a person thinks technology improves their status in society or within an organisation (Moore & Benbasat, 1991). Venkatesh and Davis (2000) and Venkatesh et al. (2003) mentioned that SI can significantly predict BI. Recently Alomari and Abdullah (2023) and Zacharis and Nikolopoulou (2022) demonstrated SI positively influencing BI. Kipreos (2019) and Lambert (2011) found that SI positively influenced BI in software development. Qualitative responses from this study also indicated that professionals were influenced by peers, social media, competitors and IT conferences. These observations served as a catalyst for the generation of a set of hypotheses where the null ($H14_0$) and alternate hypotheses ($H14_a$) are stated as:

$H14_0$: *SI does not influence the BI of software professionals to use DevOps.*

$H14_a$: *SI positively influences the BI of software professionals to use DevOps.*

6.3.7 Habit

Habit was identified as a crucial factor influencing behaviour and predicting technology usage in UTAUT2. Habit was defined by Limayem et al. (2007) "as the extent to which people tend to perform behaviours automatically because of learning". This results in executing behaviour with minimal mental effort (Wood et al., 2002). Experience with technology also plays a role in users forming a habit of using technology (Venkatesh et al., 2003). The degree of an individual's habituation to technology is contingent upon the interaction and familiarity an individual develops with a particular technology (Venkatesh et al., 2012). Hidayat et al. (2023) and Raman et al. (2022) recently demonstrated habit positively impacting BI. Participants from this study indicated that due to the benefits of DevOps, the collaborative culture and continuous improvement that it generated, it became the default strategy in their working environment. These observations served as a catalyst for the generation of a set of hypotheses where the null ($H15_0$) and alternate hypotheses ($H15_a$) are stated as:

$H15_0$: *Habit does not influence the BI of software professionals to use DevOps.*

$H15_a$: *Habit positively influences the BI of software professionals to use DevOps.*

6.3.8 Hedonic Motivation

Hedonic motivation is the enjoyment or delight of utilising a technology (Brown & Venkatesh, 2005). Venkatesh et al. (2012) suggested that hedonic motivation significantly influences the adoption and utilisation of technologies. Furthermore, Zhu and Liu (2012) mentioned that if the enjoyment of technology is high, the intention to use the technology will also increase. Marto et al. (2023) and Pinto et al. (2022) demonstrated that hedonic motivation influenced BI.

Qualitative findings from this study indicated that participants enjoyed using DevOps. This enjoyment was due to project success, collaboration, improved job satisfaction, reduced stress levels, and constant learning of new tools and technologies. This outcome served as a catalyst for the generation of a set of hypotheses where the null ($H16_0$) and alternate hypotheses ($H16_a$) are stated as:

$H16_0$: *Hedonic motivation does not influence the BI of software professionals to use DevOps.*

$H16_a$: *Hedonic motivation positively influences the BI of software professionals to use DevOps.*

6.4 REVISED CONCEPTUAL MODEL

The extensive literature review on success and acceptance factors shown in this chapter and incorporating the qualitative findings led to the presentation of a revised conceptual model, illustrated in Figure 6.1. The preliminary conceptual model was extended to include two more aspects, namely cloud computing and security, based on the qualitative comments provided by the individuals who participated in this research. In addition, the qualitative analysis alluded to links between technologies and performance expectancy and between automation and performance expectancy. Participants mentioned that technologies, as well as automation, contributed to the usefulness of DevOps. These observations served as a catalyst for the generation of a set of hypotheses where the null ($H17_0$ and $H18_0$) and alternate hypotheses ($H17_a$ and $H17_a$) are stated as:

$H17_0$: *Technologies do not influence PE in DevOps*

$H17_a$: *Technologies positively influence PE in DevOps*

$H18_0$: *Automation does not influence PE in DevOps.*

$H18_a$: *Automation positively influences PE in DevOps.*

New correlations were also developed between technologies and effort expectancy and between automation and effort expectancy. This was because participants implied that technologies and automation made DevOps easier to use. These observations served as a catalyst for the generation of a set of hypotheses where the null ($H19_0$ and $H20_0$) and alternate hypotheses ($H19_a$ and $H20_a$) are stated as:

$H19_0$: *Technologies do not influence EE in DevOps*

$H19_a$: *Technologies positively influence EE in DevOps*

$H20_0$: *Automation does not influence EE in DevOps.*

$H20_a$: *Automation positively influences EE in DevOps.*

Furthermore, based on the researcher's intuition, a relationship is established between actual usage and DevOps success. The more DevOps professionals use DevOps, the more likely it is that DevOps will be successful. This intuitive contribution will be subjected to empirical verification with the help of a set of hypotheses where the null ($H21_0$) and alternate hypotheses ($H21_a$) are stated as:

$H21_0$: *Actual usage of DevOps does not influence DevOps' success.*

$H21_a$: *Actual usage of DevOps positively influences DevOps' success.*

The revised conceptual model that incorporates all the factors linked to the BI to use DevOps, Actual Usage of DevOps and DevOps success is presented in Figure 6.1.

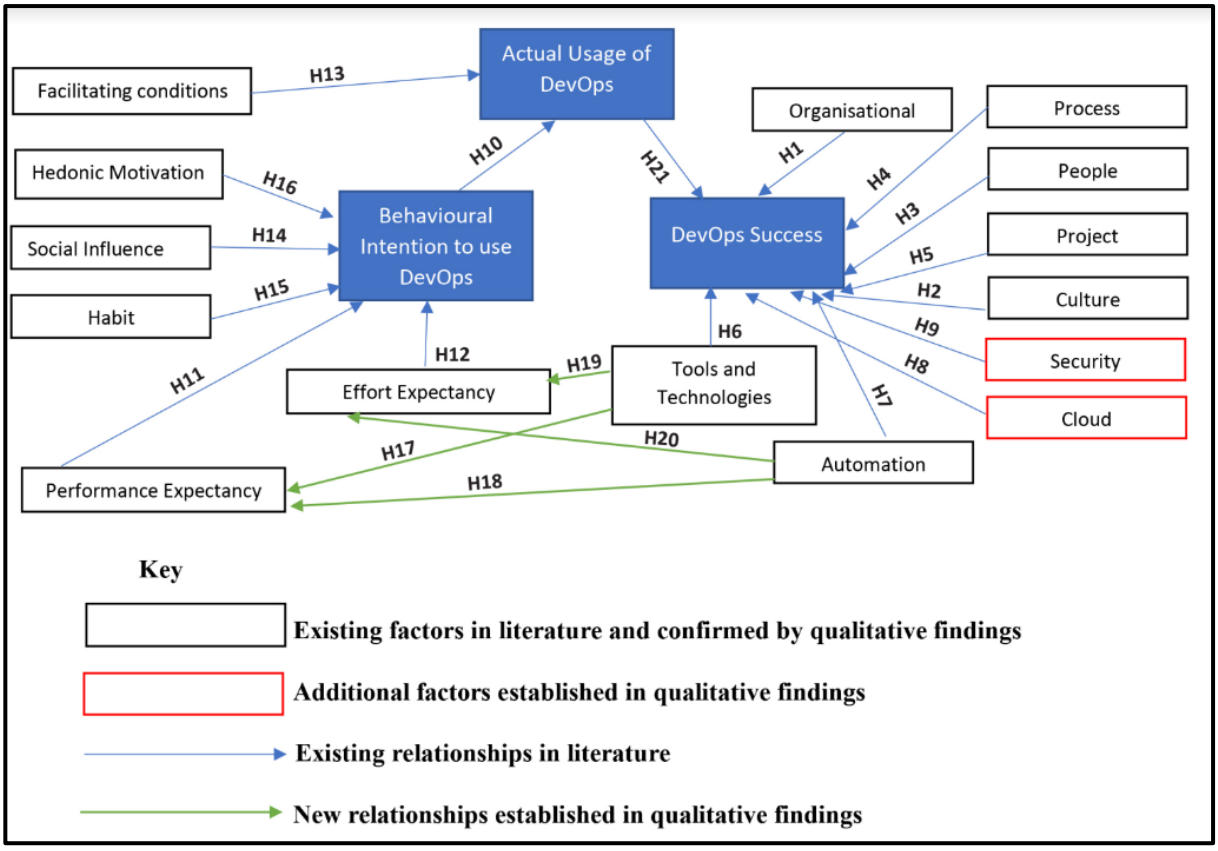

Figure 6.1: Revised Conceptual Model

6.5 SUMMARY

A revised conceptual model was established in this chapter, which was based on a comprehensive literature review and the consolidation of the qualitative findings. This resulted in the verification of the existing factors and relationships and the identification of new ones. This new model will be the blueprint for the quantitative analysis in order to determine whether or not this model is typical of the perceptions of DevOps experts in South Africa, and the results and analysis of this quantitative component are presented in the next chapter.

CHAPTER 7: QUANTITATIVE DATA ANALYSIS

7.1 INTRODUCTION

The previous chapter documented the revised conceptual model based on the literature review and the qualitative data analysis. The revised conceptual model in Figure 6.1 lays the foundation for the generation of hypotheses that are tested in the current chapter. The current chapter is guided by the following research questions:

2. What acceptance factors influence software professionals' intention to use DevOps?
3. What success factors influence DevOps project success?
4. Which acceptance and success factors are critical to the adoption of DevOps?

Research questions 2 and 3 are also repeated in this chapter since it was evaluated from both a qualitative and quantitative stance. Using a quantitative approach, DevOps professionals were invited to complete an online questionnaire that was designed to elicit their perceptions of the success and acceptance factors of DevOps.

The analysis is presented in two parts. First the descriptive statistics, which includes the demographic data and frequencies of all items and factors that affect DevOps. Second the inferential statistics, which includes factor analysis, structural equation modelling and results of the hypotheses.

7.2 DESCRIPTIVE STATISTICS

Before any data analysis, the data must be screened to ensure data quality. This means that missing data, unengaged data, and outliers that are important data quality characteristics must be investigated. Since the questionnaire was administered online and all questions were set as mandatory in order to proceed further, there was no missing data.

For unengaged responses, an overall standard deviation was calculated for all Likert scale questions. According to Collier (2020), if the standard deviation is less than 0.25, these responses could be candidates for being classified as unengaged responses. This is a reference to responses where the participants provide identical answers for most or all of the questions in the survey. Nine of the 224 responses were flagged as unengaged and deleted. Therefore, the final number of responses was 215 from the original 224.

7.2.1 Demographics Includes Respondents' Experience

The demographic details pertaining to the study's respondents are summarised in Table 7.1. This data is also presented in textual format in the sections that follow.

Table 7.1: Analysis of participant's profile

Demographic features	Classifications	Frequency	Percent	Cumulative Percent
Gender	Male	173	80.5	18.1
	Female	39	18.1	98.6
	Rather Not Mention	3	1.4	100
Age	20 - 30	75	34.9	34.9
	31- 40	92	42.8	77.7
	41-50	41	19.1	97.4
	Above 50	7	3.3	100
Tertiary Qualification	None	3	1.4	1.4
	Certificate	22	10.2	11.6
	Diploma	30	14.0	25.6
	Degree	94	43.7	69.3
	Postgraduate Degree	66	30.7	100
Type of Organisation employed in	Banking	101	47.0	47.0
	Software Development	53	24.6	71.6
	Insurance	15	7.0	78.6
	Manufacturing	7	3.3	81.9
	Telecommunications	8	3.7	85.6
	Others	31	14.4	100

- The majority of the respondents were male (80.5%). Females comprised 18.1% of respondents, with 1.4% preferring not to mention their gender.
- The 31 – 40 age group had the most respondents (42.8%), followed by the 20- 30, which had 34.9% of the respondents.
- Most of the respondent's (43.7%) highest qualifications are a degree, and 30.7% have a postgraduate degree.
- While respondents predominantly work in the banking sector (47%), a fair number (24.6%) work in software development companies.

Information pertaining to the respondents' experience in using Agile methodology as well as DevOps is summarised in Table 7.2. This data is also presented in textual format in the sections that follow.

Table 7.2: Respondents software development experience

Project Overview	Project Details	Frequency	Percent	Cumulative Percent
Agile Methodology in Organisation	Scrum	152	70.6	70.6
	Feature Driven Development	12	5.6	76.2
	Extreme Programming	13	6.0	82.2
	Lean Software Development	13	6.0	88.2
	Adaptive Software Development	9	4.2	92.4
	DSDM and Others	16	7.4	100
Project Team Size	5 or less members	33	15.3	15.3
	6 – 10 team members	78	36.3	51.6
	11 -15 team members	59	27.4	79
	More than 15 members	45	21.0	100
Project/Feature length	Less than 3 months	56	26.0	26.0
	3 – 6 months	52	24.2	50.2
	7 – 9 months	41	19.1	69.3
	10 – 12 months	23	10.7	80
	More than 12 months	43	20.0	100.0
Job Responsibility	DevOps Engineer	66	30.7	30.7
	Developer	53	24.6	55.3
	Business Analyst	6	2.8	58.1
	Operations	12	5.6	63.7
	System Analyst	10	4.7	68.4
	Project Manager	15	7.0	75.4

	Security	8	3.7	79.1
	Team Lead	22	10.2	89.3
	Testing	8	3.7	93
	Others	15	7.0	100
Experience with DevOps (years)	1-2	58	27.0	27.0
	3 - 4	73	34	61.0
	5 - 6	58	27	88.0
	7 – 8	12	5.6	93.6
	Above 8	14	6.5	100.0
Number of DevOps project/feature involvement	1 -5	70	32.6	37.1
	6 - 10	65	30.2	62.8
	11 -15	26	12.1	74.9
	16 - 20	12	5.6	80.5
	21 - 25	4	1.9	82.4
	Above 25	38	17.7	100

Type of Agile Methodology in Organisation - 70.6% of the respondents indicate that the organisation they are employed in uses the Scrum Agile development methodology. Other methodologies mentioned by respondents were Feature Driven Development (5.6%), Extreme Programming (6%), Lean Software Development (6%) and Adaptive software development (4.2%). Others (7.4%) include custom-developed Agile methodologies, using various Agile methodologies and Kanban.

Project Team Size - Respondents indicated that the project team size mainly consisted of 6 – 10 team members (36.3%). Also, 27.4% indicated a project team size of 11-15 members and 21% indicated a team size of over 15.

Project/Feature Length - Respondents indicated that the projects took on varying lengths of time. Twenty-six percent of respondents indicated the project length was less than three months, 24.2 indicated between 3 and 6 months, 19.1% indicated between 7 and 9 months, 10% indicated between 10 – 12 months, and 20% indicated more than 12 months.

Job Responsibility - The respondents were primarily DevOps engineers (30.7%) and Developers (24.6%). Other job profiles were Business Analyst (2.8%), Operations (5.6%), System Analyst

(4.7%), Project Manager (7%), Security (3.7%), Team Lead (10.2%) and Testing (3.7%). Others (7%) were CIOs, directors, project sponsors and solution architects.

Experience with DevOps - Thirty-four percent of respondents had 3 – 4 years' experience with DevOps, 27% had 1 – 2 years of experience, 27% had 5 – 6 years of experience, 5.6% had 7 – 8 years of experience, and 6.5% had more than eight years of experience.

Number of DevOps project/feature involvement - 32.6% of the respondents indicated they were involved in 1-5 DevOps projects, 30.2% in 6-10 DevOps projects, and 12.1% in 11-15 DevOps projects, with 17.7% involved in over 25 DevOps projects.

7.2.2. Acceptance Factors

The software professionals' level of agreement with different items that make up each construct is provided. Thereafter sentiment analysis is performed by converging each 5-point Likert scale used in the computation for the different items that make up the acceptance factors into 3 categories, where strongly disagree and disagree was conflated as the negative category, neutral remaining neutral and strongly agree and agree was conflated as the positive category.

Performance Expectancy

Performance expectancy (PE) sought to establish how useful DevOps has been towards the success of software development. Figure 7.1 represents the software professional's level of agreement with different items that make up the PE construct.

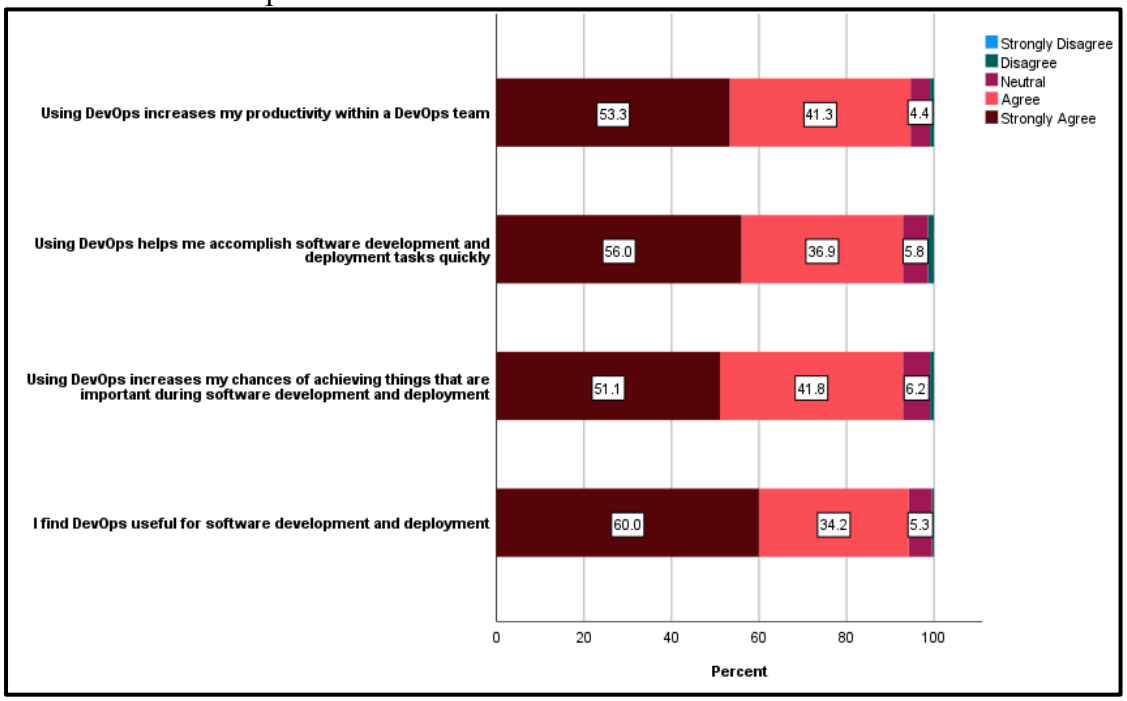

Figure 7.1: Frequency of Performance Expectancy Construct

Figure 7.1 shows that over 90% of the respondents either agreed or strongly agreed that DevOps positively impacted performance during software development. The items "*Using DevOps increases my productivity within a DevOps team*" and "*I find DevOps useful for software development and deployment*" had the most positive responses. The mean was recorded at 4.48, the standard deviation (SD) = 0.58, and the median = 4.75 (*M*=4.48; *SD* = 0.58; *Mdn*= 4.75)

This observation is corroborated by the empirical evidence provided in the Sentiment analysis (Figure 7.2).

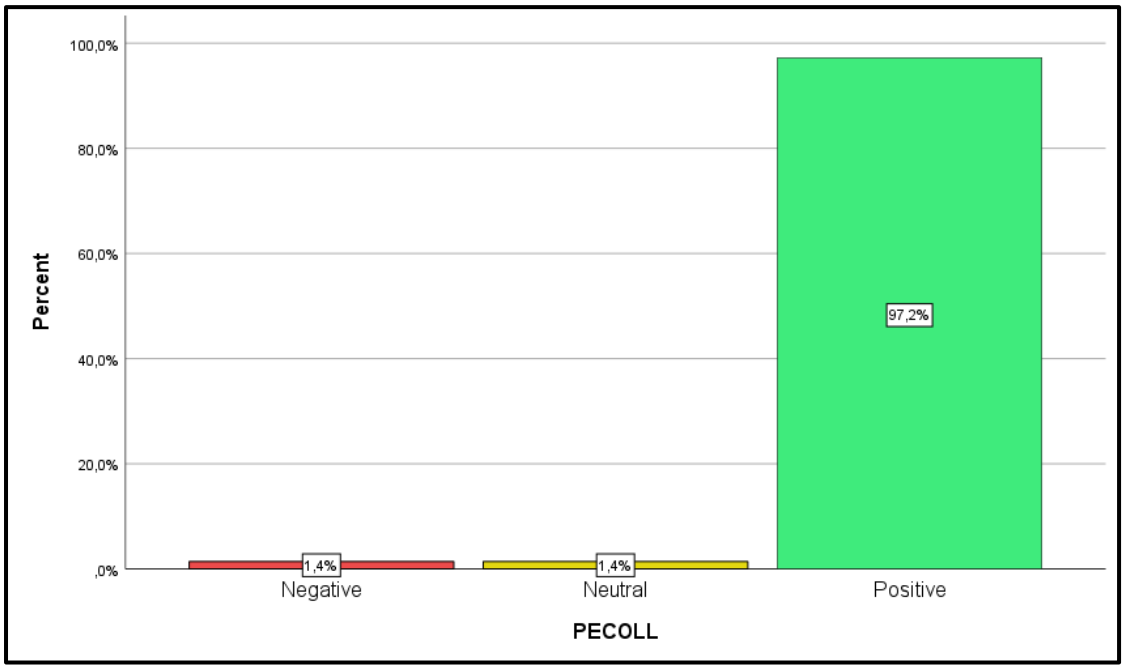

Figure 7.2: Sentiment Analysis of the Performance Expectancy Construct

The evidence shown in Figure 7.2 suggest an overwhelming **positive** disposition of the respondents (97.2%, n=215) that is suggestive of the outcome that *Performance Expectancy* positively influences their sentiment towards using DevOps.

In order to establish whether these aggregated values (*M*=4.48; *SD* = 0.58; *Mdn*= 4.75) pertaining to the PE construct was statistically significant or purely by chance, the data was subjected to a test of normality, the outcome of which will enable the identification of an accurate test for statistical significance. According to Field (2005)), the Kolmogorov-Smirnov or Shapiro-Wilk tests can be used to test normality and a non-significant result ($p>0.05$) indicates the normal distribution of data, and a significant result ($p<0.05$) indicates a non-normal distribution. As illustrated in Table 7.3, both these tests yield significant results ($p<0.05$) thereby suggesting that the data has a non-normal distribution.

Table 7.3: Test of Normality

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Statistic	**df**	**Sig.**	**Statistic**	**df**	**Sig.**
Organisational	,071	215	,010	,985	215	,019
Security	,123	215	<,001	,956	215	<,001
People	,079	215	,003	,983	215	,012
Process	,095	215	<,001	,965	215	<,001
Project	,134	215	<,001	,972	215	<,001
Technologies	,120	215	<,001	,924	215	<,001
Automation	,161	215	<,001	,917	215	<,001
Cloud	,142	215	<,001	,940	215	<,001
Culture	,142	215	<,001	,904	215	<,001
Success	,137	215	<,001	,942	215	<,001
Performance Expectancy	,263	215	<,001	,805	215	<,001
Effort Expectancy	,129	215	<,001	,948	215	<,001
Facilitating Conditions	,164	215	<,001	,925	215	<,001
Hedonic Motivation	,201	215	<,001	,872	215	<,001
Habit	,133	215	<,001	,913	215	<,001
Social Influence	,166	215	<,001	,913	215	<,001
Behavioural Intention	,184	215	<,001	,857	215	<,001

The non-normal data distribution necessitates the use of a non-parametric statistical significance testing strategy. The Wilcoxon-one sample signed ranked test was chosen instead of the parametric t-test, due to the non-normal data distribution, as indicated in Table 7.3. The Wilcoxon Test uses a hypothesised median value of 3 as the main comparison test statistic. The results of the Wilcoxon Test (fully tabulated in Appendix D Table D.1) is graphically illustrated in Figure 7.3 where it shows that the observed median is significantly different from the hypothesised median.

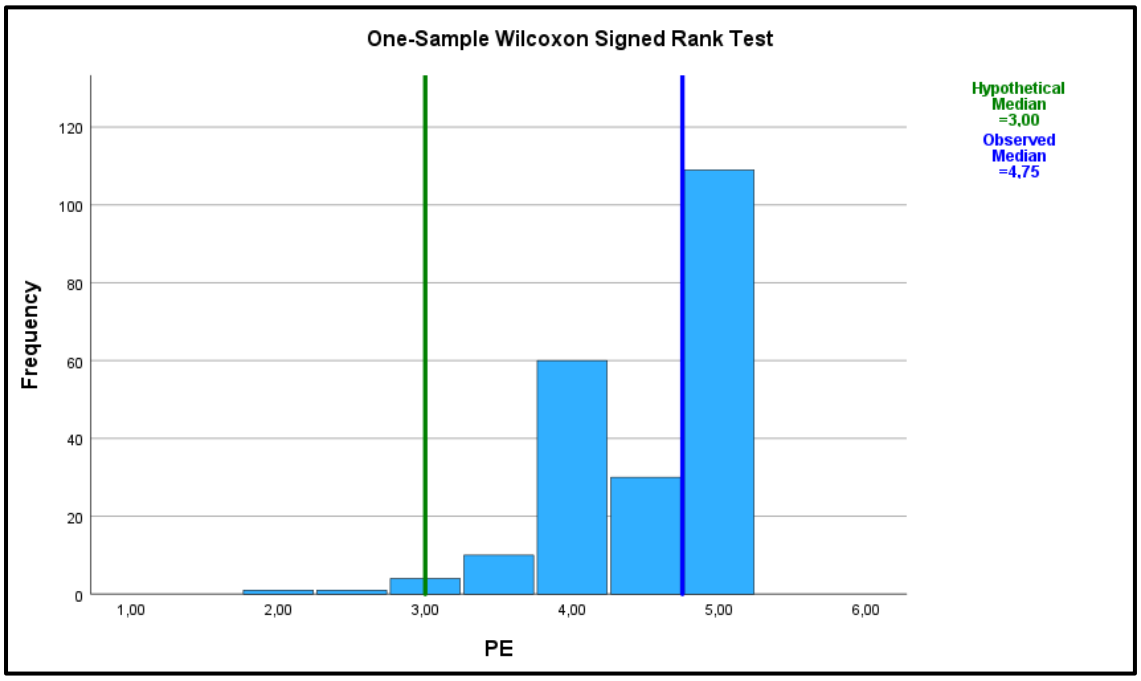

Figure 7.3: Central Tendency Data for Performance Expectancy Construct

Effort Expectancy

Effort expectancy (EE) sought to establish how easy it was for software professionals to use DevOps. Figure 7.4 represents the software professionals' level of agreement with different items that made up the EE construct. The figure illustrates that over 65% of the respondents either agreed or strongly agreed that DevOps was easy to use. The items "*It is easy for me to become skillful in using DevOps*", "*I find DevOps easy to use when developing and deploying software*" and "*My interaction with DevOps is clear and understandable*" had the most positive agreement responses with 72.4%, 77.4% and 76.4% respectively.

For the EE construct the mean was observed to be 3.94, standard deviation (SD) = 0.72, and median = 4.0 (*M*=3.94; *SD* = 0.72; *Mdn*= 4.0).

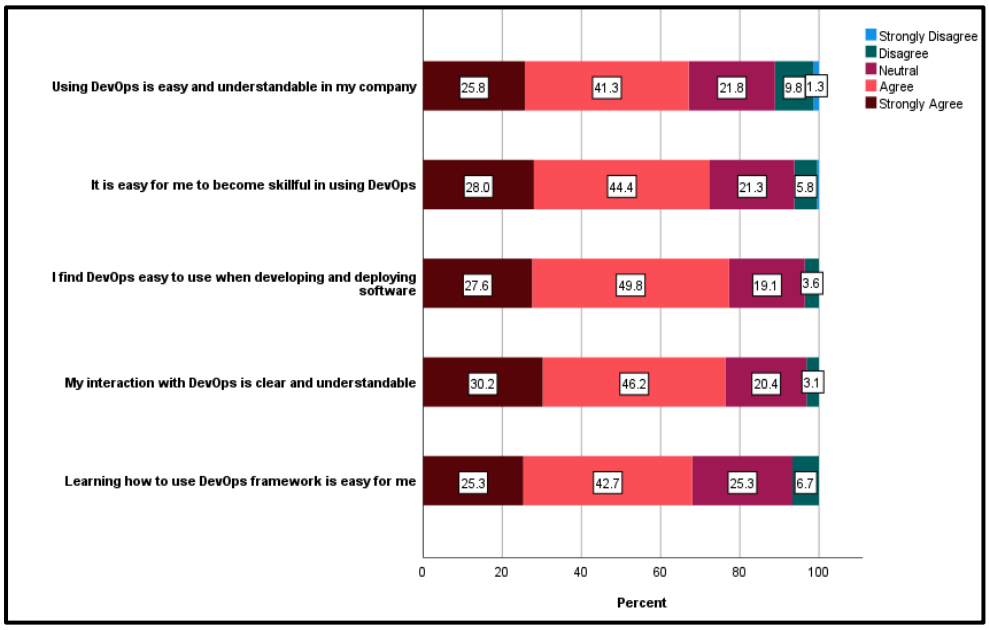

Figure 7.4: Frequency of the Effort Expectancy Construct

The conflated view, presented in Figure 7.5 shows that the majority (87%, n=215) of the respondents have a positive perception towards EE and, therefore, perceive DevOps as easy to use and learn.

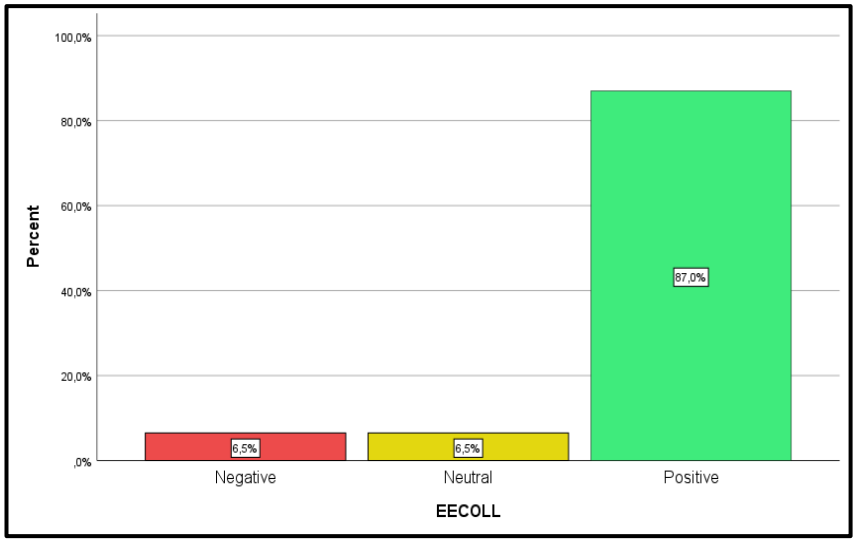

Figure 7.5: Sentiment Analysis of the Effort Expectancy Construct

The Wilcoxon signed test was used to test if the positive perception of the construct of EE is statistically significant. The results of the Wilcoxon Test (fully tabulated in Appendix D, Table D.1) is graphically illustrated in Figure 7.6 where it is shown that the observed median is significantly different from the hypothesised median. This suggests that there is significant agreement among software professionals that they were able to use and learn DevOps with little effort.

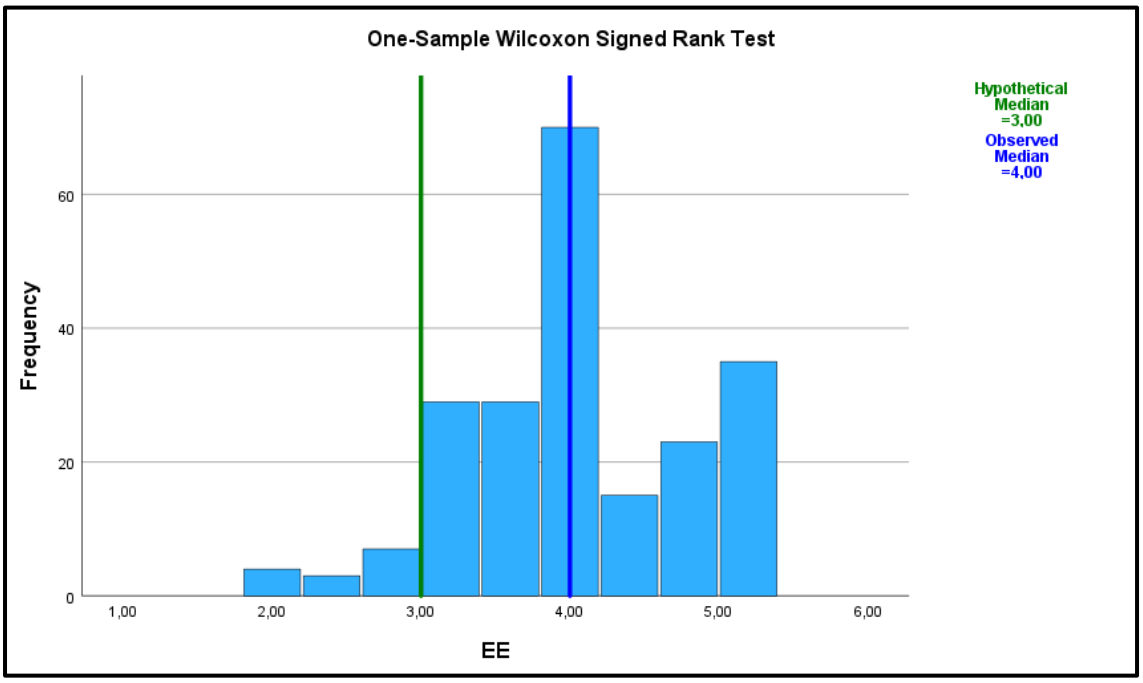

Figure 7.6: Central Tendency Data for Effort Expectancy Construct

Facilitating Conditions

Facilitating Conditions (FC) sought to establish whether DevOps professionals have the necessary resources and knowledge to use the DevOps strategy. Figure 7.7 represents the software professionals' level of agreement with different items that made up the FC construct. The figure illustrates that over 76% of the respondents either agreed or strongly agreed that DevOps was compatible with their other software development processes. Furthermore, 79.6% of respondents indicated that they have the knowledge necessary to use DevOps strategy for software development, and 82.7% implied that they had the necessary resources to use it.

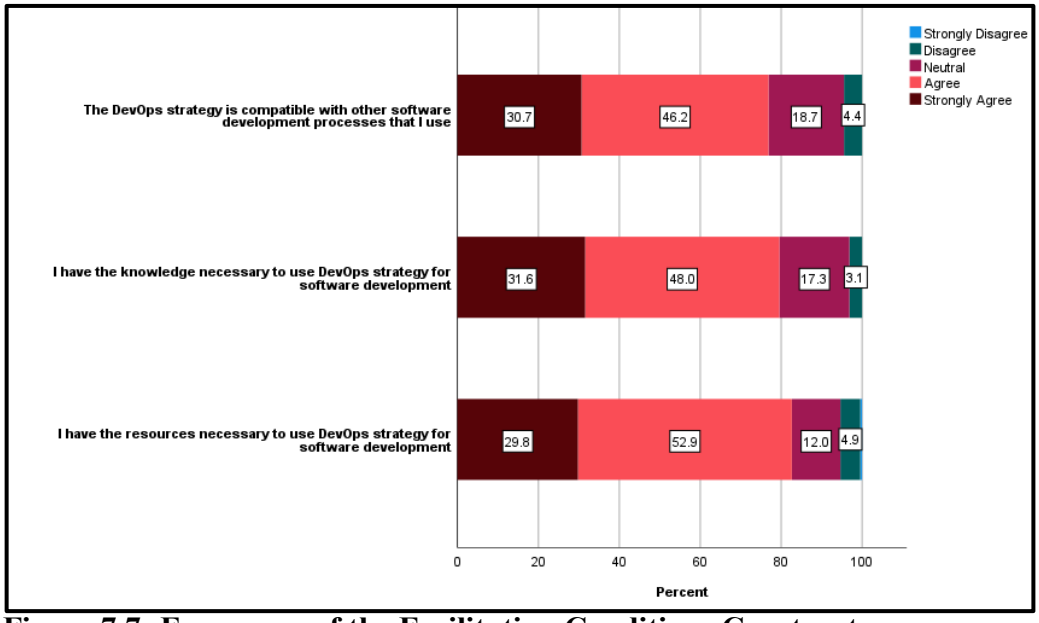

Figure 7.7: Frequency of the Facilitating Conditions Construct

For the FC construct, the mean was observed to be 4.06, standard deviation (SD) = 0.69, and median = 4.0 (*M*=4.06; *SD* = 0.69; *Mdn*= 4.0).

The conflated view, presented in Figure 7.8 shows the majority (88.4%, n=215) of the respondents have a positive perception towards FC, and therefore, DevOps professionals perceive they have the resources and knowledge to use DevOps.

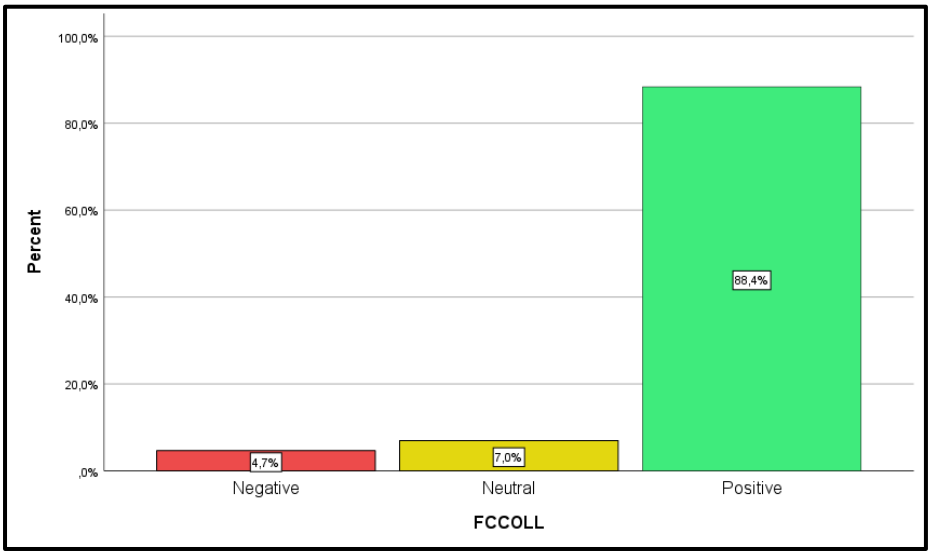

Figure 7.8: Sentiment Analysis of the Facilitating Conditions Construct

The Wilcoxon signed test was used to test if the positive perception of the construct of FC is statistically significant. The results of the Wilcoxon Test (fully tabulated in Appendix D, Table D.1) is graphically illustrated in Figure 7.9 where it is shown that the observed median is significantly

different from the hypothesised median. This suggests that there is significant agreement among software professionals that they have the necessary facilitating conditions for DevOps.

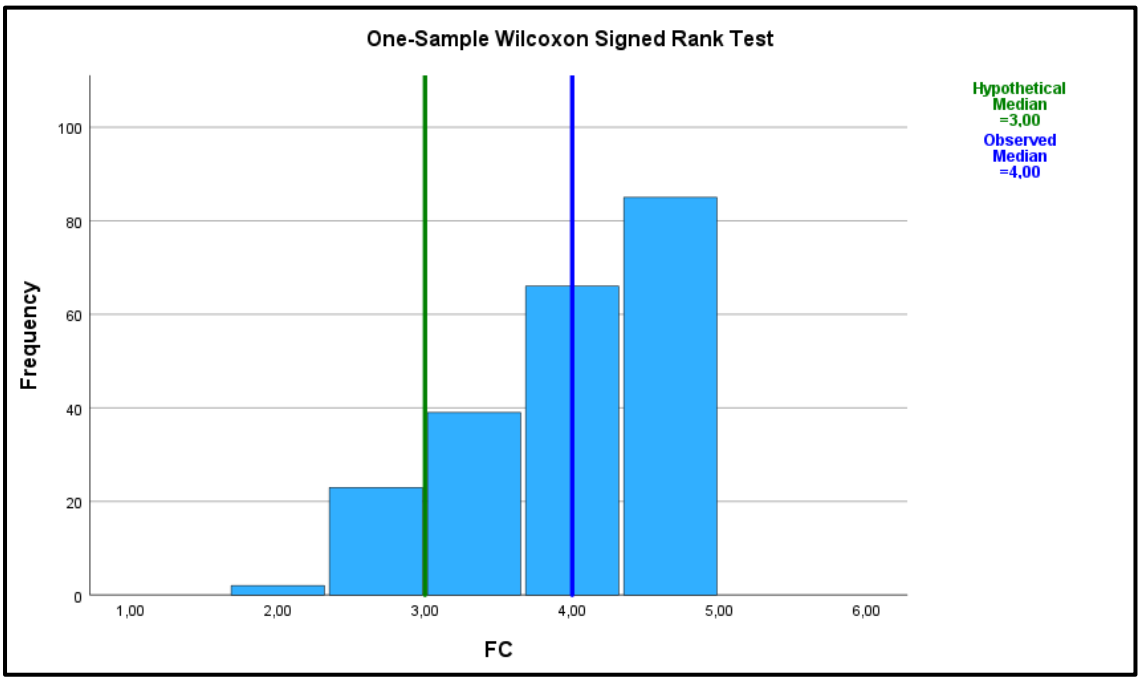

Figure 7.9: Central Tendency Data for the Facilitating Conditions Construct

Social Influence

Social Influence (SI) sought to establish whether others have influenced the acceptance of DevOps by software professionals. Figure 7.10 represents the software professionals' level of agreement with different items that made up the SI construct. The figure illustrates that 77.8% of the respondents either agreed or strongly agreed that people whose opinions they value prefer them to use DevOps for software development. Furthermore, 74.2% of respondents indicated that people who influence their behaviour think they should use DevOps for software development, and 66.6% of the respondents implied that people who are important to them think they should use DevOps for software development.

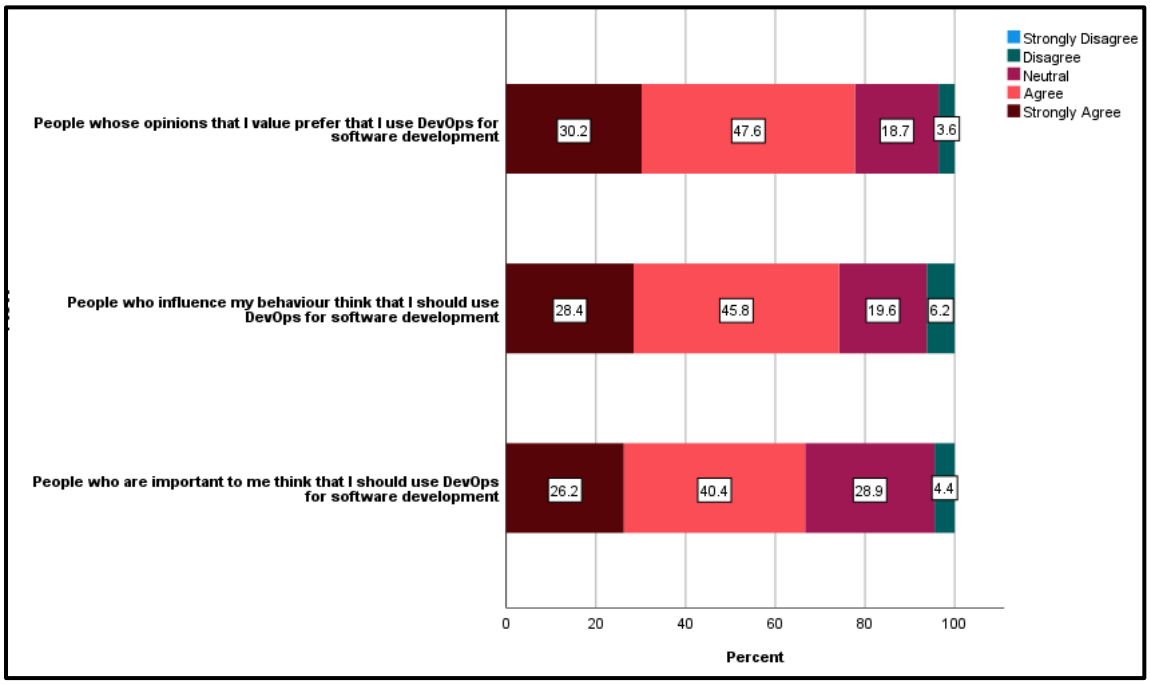

Figure 7.10: Frequency of the Social Influence Construct

For the SI construct, the mean was 3.96, standard deviation (SD) = 0.78, and median = 4.0, which indicates that most respondents agreed or strongly agreed that people influence their use of DevOps (*M*=3.96; *SD* = 0.78; *Mdn*= 4.0).

The conflated view, presented in Figure 7.11 shows that the majority (80.9%, n=215) of the respondents have a positive perception towards SI, and therefore, DevOps professionals perceive that people whose opinions they value influence their behaviour and think they should use DevOps.

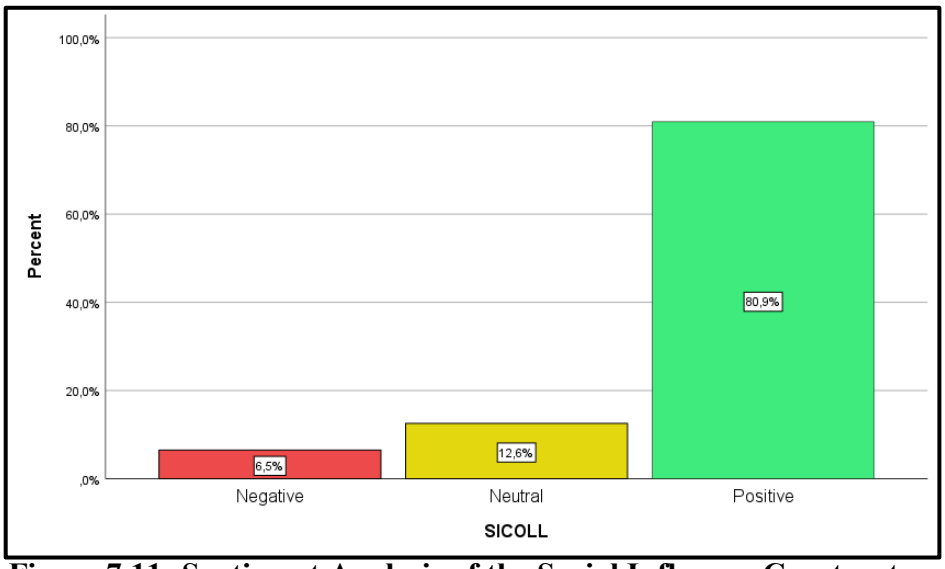

Figure 7.11: Sentiment Analysis of the Social Influence Construct

The Wilcoxon signed test was used to test if the positive perception of the construct of SI shown in Figure 7.11 is statistically significant. The results of the Wilcoxon Test (fully tabulated in Appendix D Table D.1) is graphically illustrated in Figure 7.12 where it is shown that the observed median is significantly different from the hypothesised median

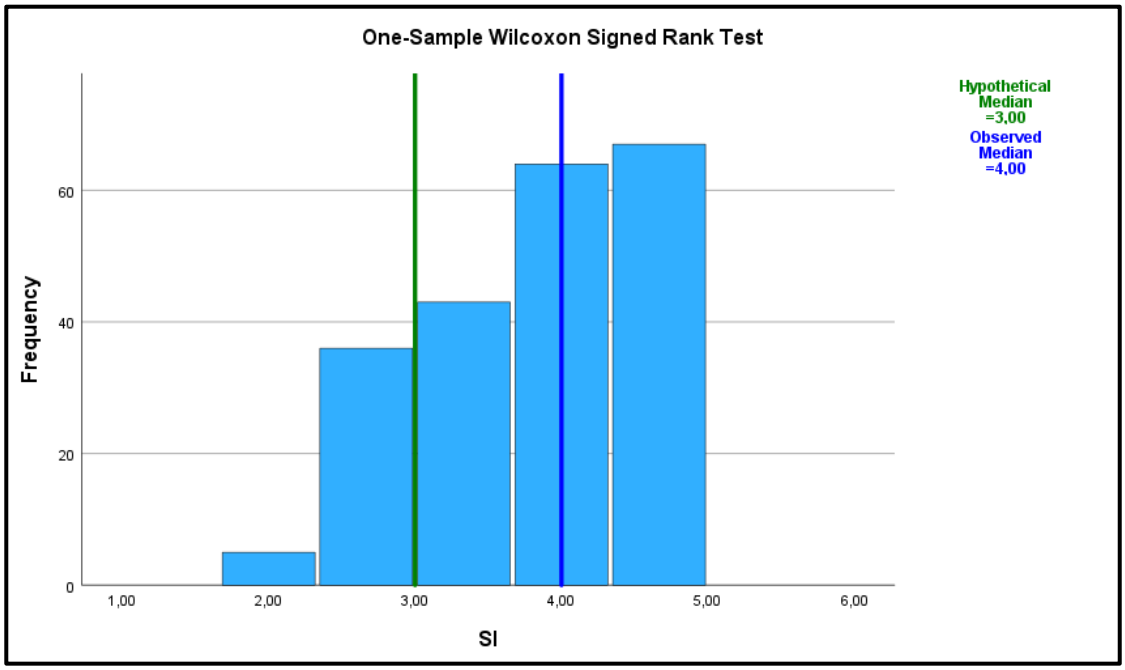

Figure 7.12: Central Tendency Data for Social Influence Construct

This illustration shown in Figure 7.12 suggests that there is significant agreement among software professionals that their use of DevOps is endorsed by their peers thereby generating a positive disposition towards the SI construct in the context of DevOps use.

Habit

Habit sought to establish whether software professionals will spontaneously use the DevOps strategy. Figure 7.13 represents the software professionals' level of agreement with different items that made up the Habit construct. The figure illustrates that over 89.7% of the respondents either agreed or strongly agreed that it became a habit for them to research new tools and technologies. Furthermore, 88% of respondents mentioned that it became a habit to interact with team members in other domains to fix projects when it fails, and 86.7% mentioned that looking for ways to automate their software development processes became a habit. 84.9% indicated that using DevOps became natural to them, and 84% suggested it became a habit to collaborate with team members from other IT domains.

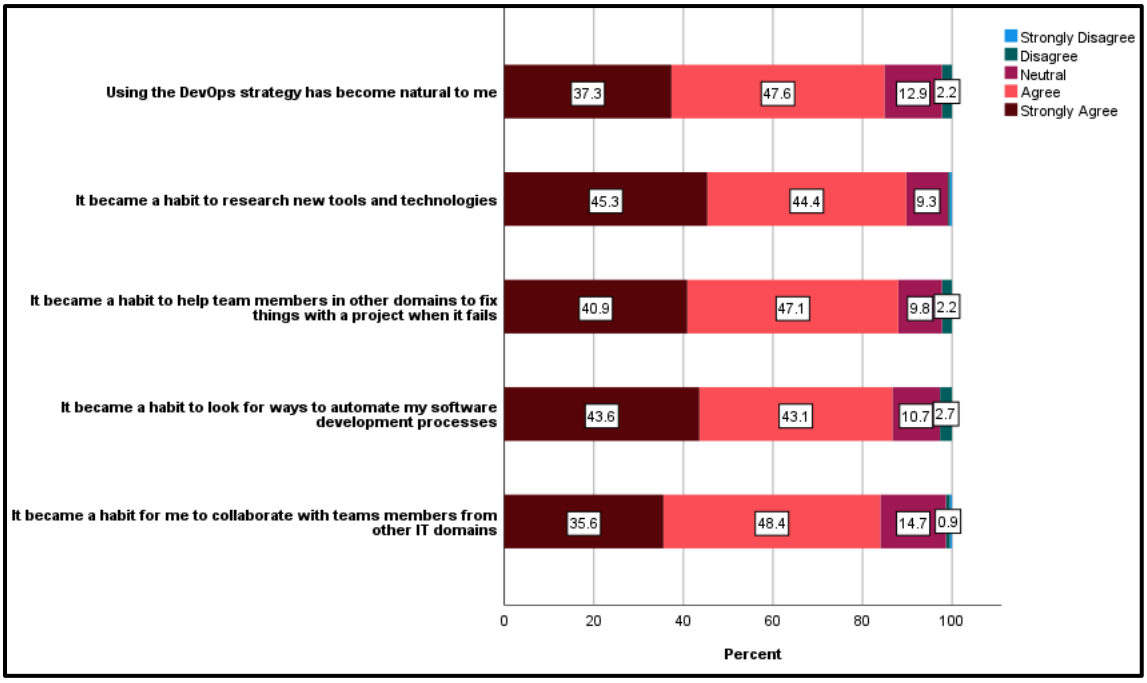

Figure 7.13: Frequency of the Habit Construct

For the Habit construct, the mean was recorded as 4.3, standard deviation (SD) = 0.61, and median = 4.2, which indicates that most respondents agreed or strongly agreed that DevOps became a habit for them (*M*=4.3; *SD* = 0.61; *Mdn*= 4.2).

The conflated view, presented in Figure 7.14 shows the majority (97.2%, n=215) of the respondents have a positive perception towards the Habit construct, indicating DevOps became a habit for them.

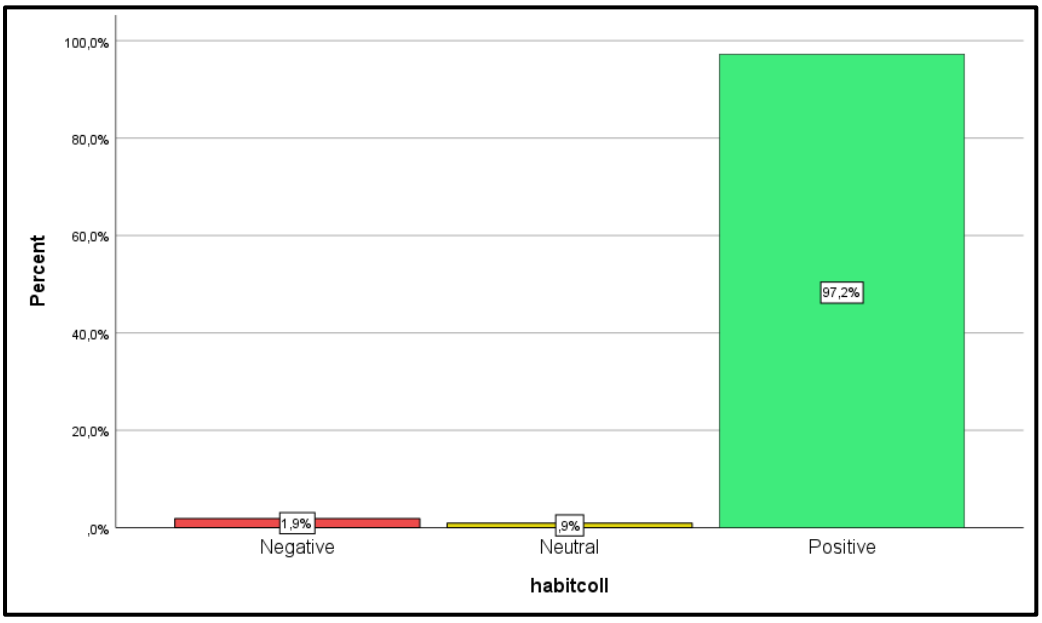

Figure 7.14: Sentiment Analysis of the Habit Construct

The Wilcoxon signed test was used to test if the positive perception of the construct of Habit shown in Figure 7.14 is statistically significant. The results of the Wilcoxon Test (fully tabulated in Appendix D, Table D.1) is graphically illustrated in Figure 7.15 where it is shown that the observed median is significantly different from the hypothesised median thereby indicating that there was a significant agreement among software professionals that using DevOps became a habit for them.

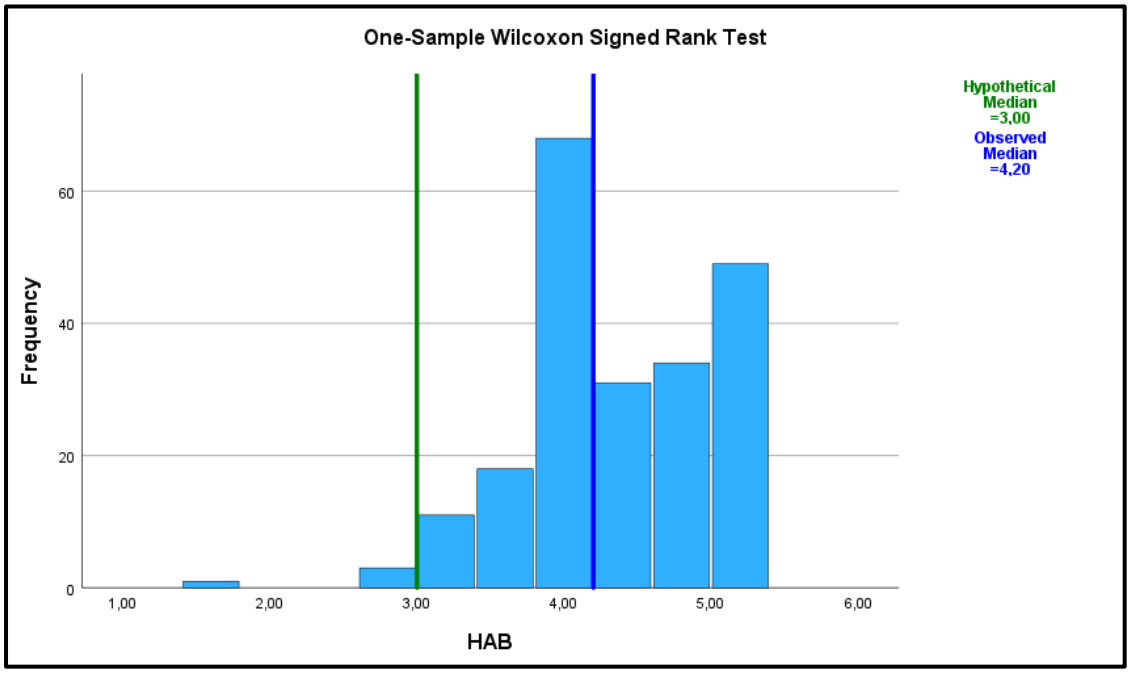

Figure 7.15: Central Tendency Data for the Habit Construct

Hedonic motivation

Hedonic Motivation (HM) sought to establish whether software professionals' positive experience with DevOps will result in the use of DevOps. Figure 7.16 represents the software professionals' level of agreement with different items that made up the HM construct. The figure illustrates that over 95.2% of the respondents either agreed or strongly agreed that they enjoyed learning from team members in other domains. Furthermore, 93.8% of respondents agreed or strongly agreed that they enjoy interaction with team members in other domains, 92% agreed or strongly agreed that they have fun playing with and investigating new tools to make the DevOps strategy better, and 90.6% agreed or strongly agreed that they are motivated to work since things get done faster and problems are solved more efficiently and effectively.

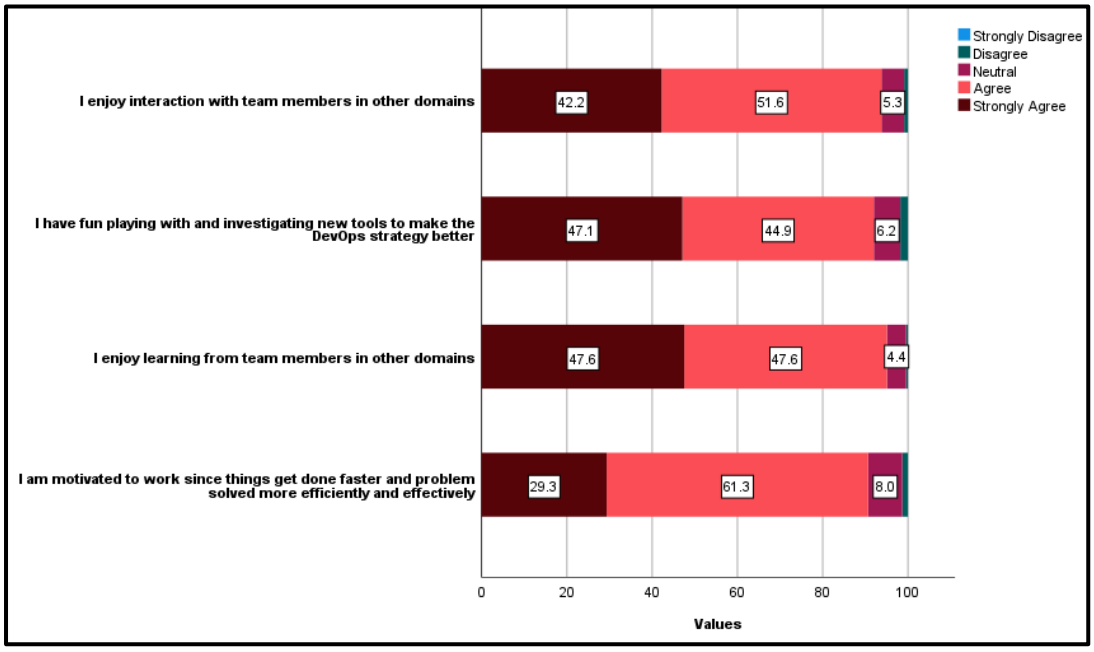

Figure 7.16: Frequency of the Hedonic Motivation Construct

For the HM construct, the mean was 4.33, standard deviation (SD) = 0.53, and median = 4.25, indicating that most respondents agreed or strongly agreed that they had a positive experience with using DevOps (*M*=4.3; *SD* = 0.53; *Mdn*= 4.25). The conflated view, presented in Figure 7.17 shows the majority (97.7%, n=215) of the respondents have a positive perception towards the HM construct, indicating that they are more likely use DevOps since they will be motivated by its positive experience.

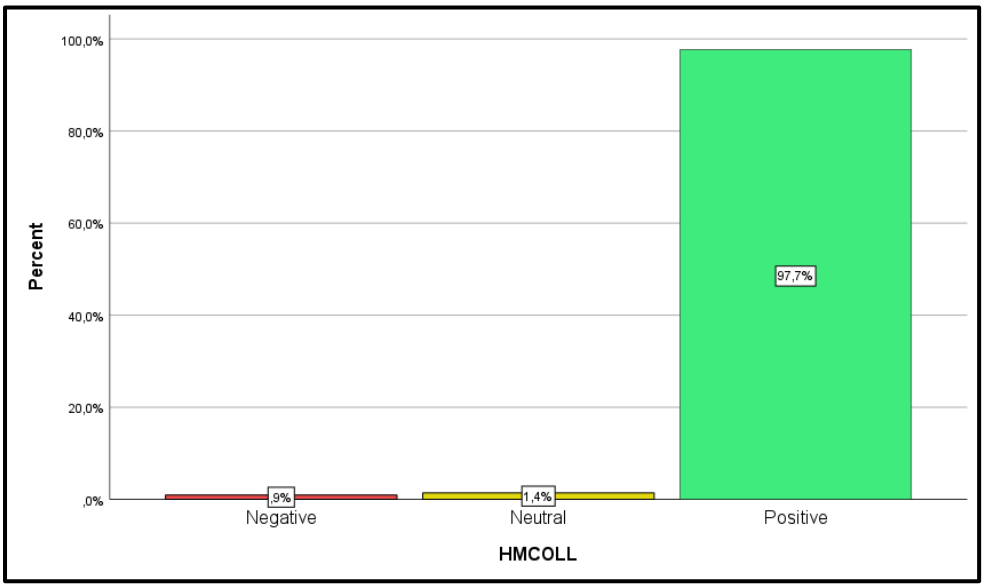

Figure 7. 17: Sentiment Analysis of the Hedonic Motivation Construct

The Wilcoxon signed test was used to test if the positive perception of the construct of HM shown in Figure 7.17 is statistically significant. The results of the Wilcoxon Test (fully tabulated in Appendix D Table D.1) is graphically illustrated in Figure 7.18 where it is shown that the observed median is significantly different from the hypothesised median thereby indicating that there was a significant agreement among software professionals that using DevOps was enjoyable and fun and motivates them to continue using it.

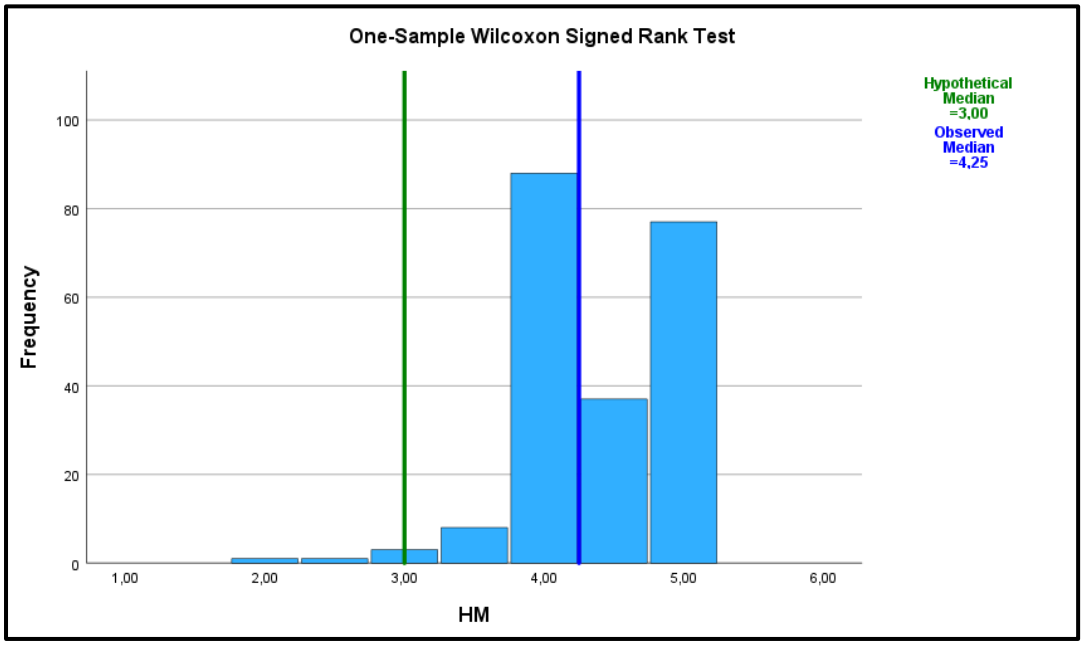

Figure 7.18: Central Tendency Data for Hedonic Motivation Construct

Behavioural Intention

BI sought to establish whether software professionals will continue to use DevOps. Figure 7.19 represents the software professionals' level of agreement with different items that made up the BI construct. The figure illustrates that about 94% of the respondents either agreed or strongly agreed with the items "*I predict that I will be able to use a DevOps approach for software development projects in my organisation*", "Based on my knowledge of DevOps methodology, I predict that I would make an effort to use it more often", "*I intend to make an effort to practice a DevOps approach for software development projects on a regular basis*" and "*I intend to use a DevOps approach for software development projects on a regular basis*".

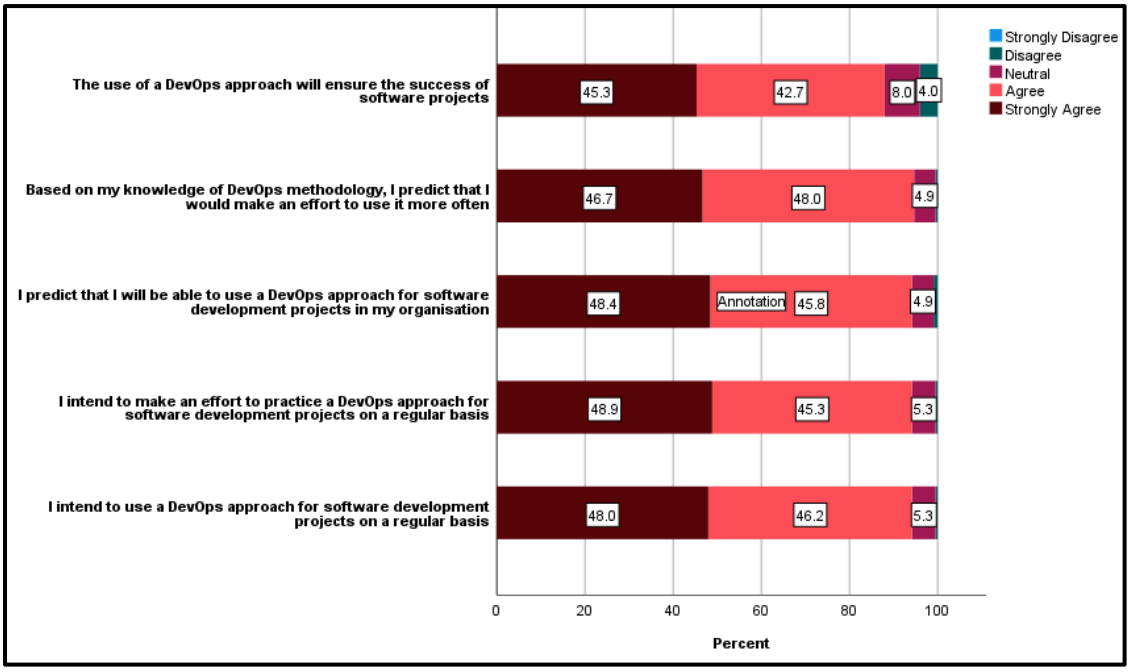

Figure 7.19: Frequency of the Behavioural Intention Construct

For the construct of BI, the mean was recorded at 4.4, standard deviation (SD) = 0.56, and median = 4.4, indicating that most respondents agreed or strongly agreed to use DevOps (*M*=4.3; *SD* = 0.53; *Mdn*= 4.25). The conflated view, presented in Figure 7.20 shows the majority (98.1%, n=215) perceive the BI construct positively thereby indicating a general intent to use DevOps.

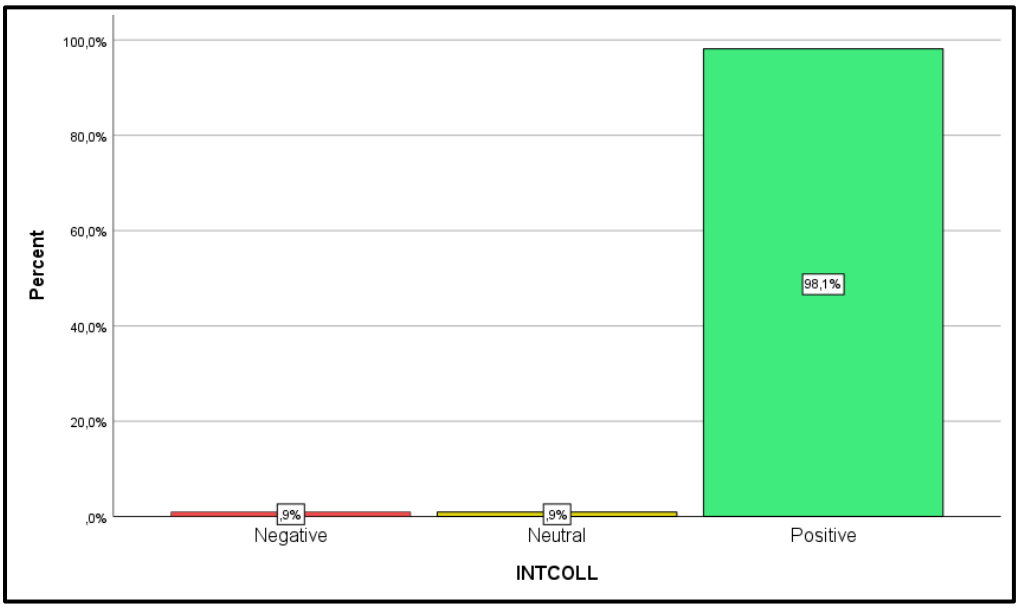

Figure 7.20: Sentiment Analysis of the Behavioural Intention Construct

The Wilcoxon signed test was used to test if the positive values recorded for the construct of BI was statistically significant. The results of the Wilcoxon Test (fully tabulated in Appendix D, Table D.1) is graphically illustrated in Figure 7.21 where it is shown that the observed median is significantly different from the hypothesised median thereby indicating that there was a showed a significant agreement among software professionals that they intend to use DevOps.

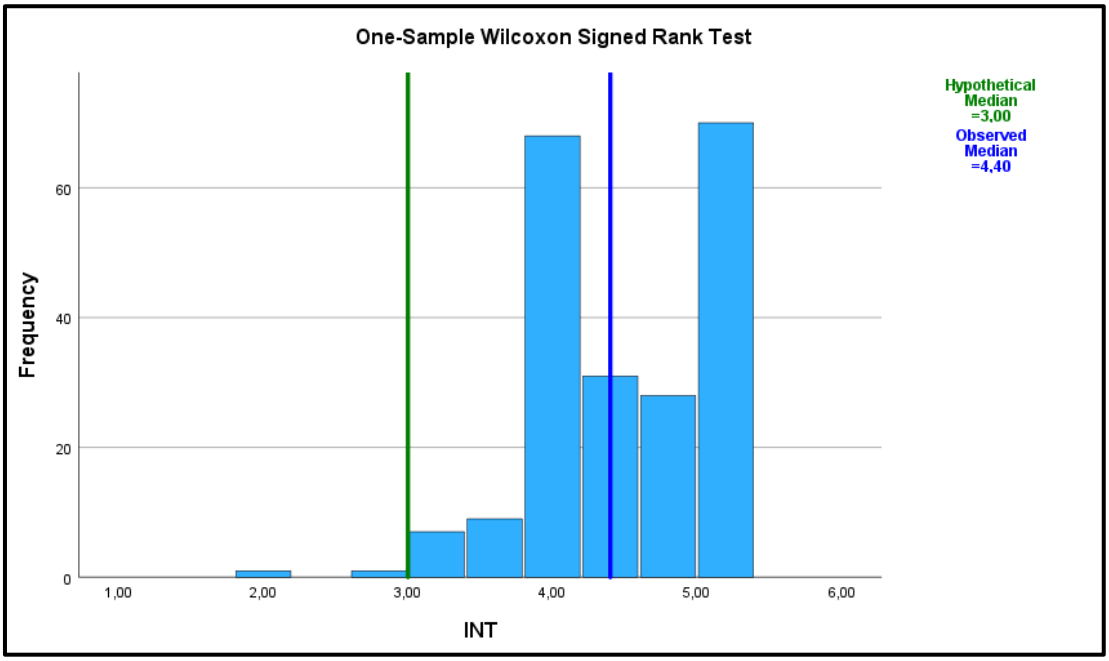

Figure 7.21: Central Tendency Data for Behavioural Intention Construct

Actual Usage

The descriptive statistics presented thus far were based on perceptions and BI, the actual usage of DevOps was also recorded. Figure 7.22 shows that 46.51% of the respondents indicated that they "*almost always*" use DevOps, 30.23% use DevOps *often*, 18.14% *sometimes* and 5.12% *seldom*.

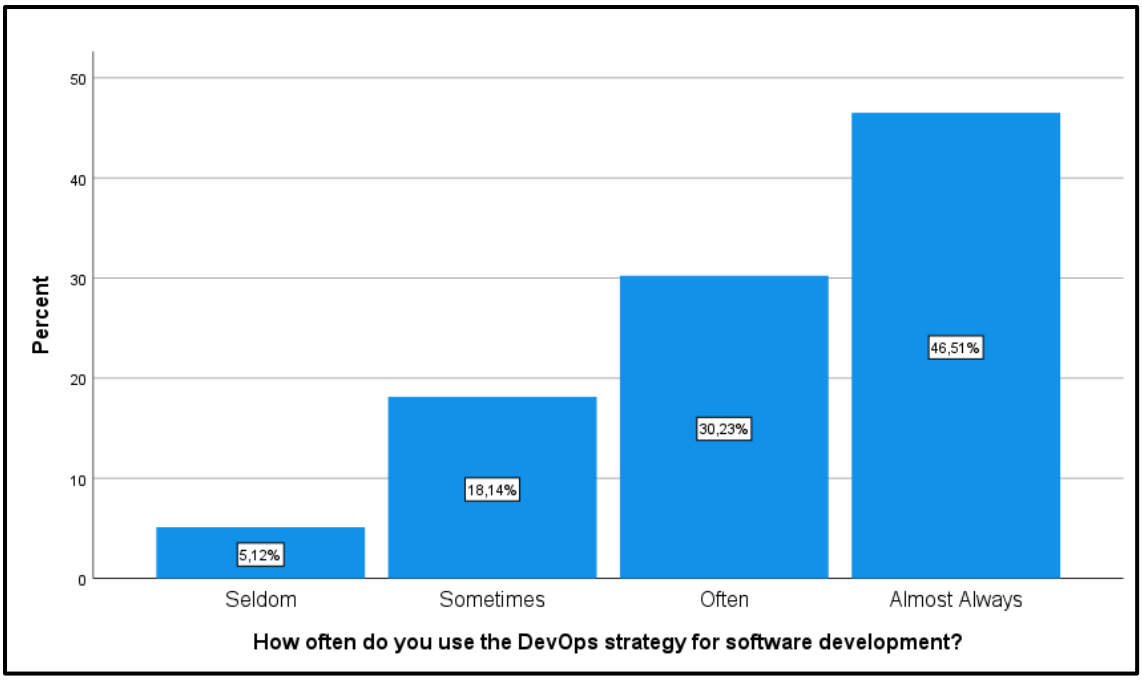

Figure 7.22: Frequency of the Actual Usage Construct

7.2.3 Success Factors

The software professionals' level of agreement with different items that make up each construct is provided. Thereafter sentiment analysis is performed by converging each 5-point Likert scale used in the computation for the different items that make up the success factors into 3 categories, where strongly disagree and disagree was conflated as the negative category, neutral remaining neutral and strongly agree and agree was conflated as the positive category.

Organisational

Organisational factors sought to establish how the organisational practices and people behave in a DevOps environment to influence its success. Figure 7.23 represents the software professionals' level of agreement with different items that make up the organisational construct. This figure shows that 78.7% of the respondents either agreed or strongly agreed that DevOps project team received good management or executive support, 76% agreed or strongly agreed that the DevOps project team has a committed sponsor or a committed organisation manager, 72.4% agreed or strongly agreed that the organisation has a cooperative culture instead of hierarchical. Only 41.3% agreed or strongly agreed that the organisation has a reward system that recognises DevOps behaviour, 53.8% agreed

or strongly agreed that the organisation has an oral culture that places a high value on face-to-face communication style. Also, 39.6% agreed or strongly agreed that the DevOps team is collocated, resulting in team members working in the same location for ease of communication and casual and constant contact.

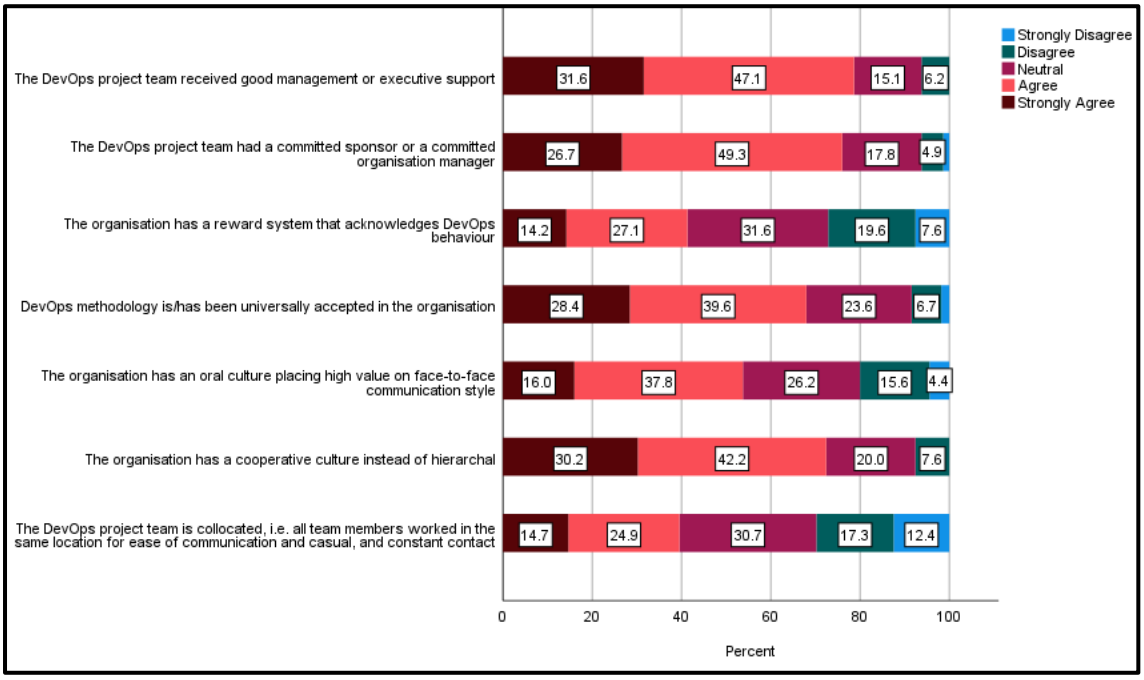

Figure 7.23: Frequency of the Organisational Construct

For the Organisational construct the mean was recorded as 3.67, the standard deviation (SD) =0.61, and the median = 3.71indicating that most respondents agreed or strongly agreed to use DevOps (*M*=3.67; *SD* = 0.61; *Mdn*= 3.7). The conflated view, presented in Figure 7.24 shows the majority (85.1%, n=215) perceive the organisational construct positively.

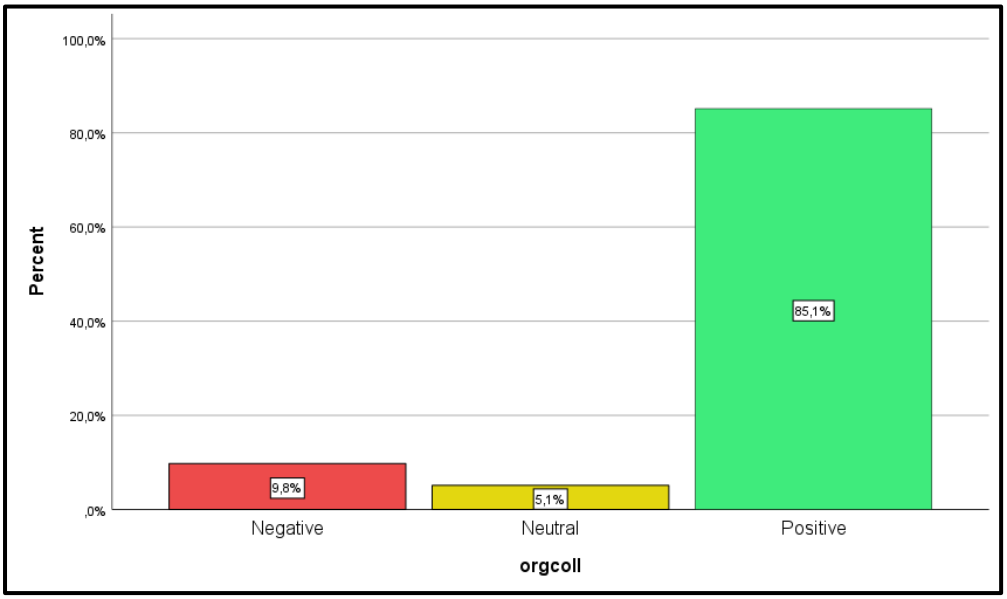

Figure 7.24: Sentiment Analysis of the Organisational Construct

These results suggest that most respondents agreed or strongly agreed that the organisation has implemented appropriate organisational factors to enable the success of DevOps projects.

The Wilcoxon signed test was used to test if the positive values recorded for the Organisational construct in the context of DevOps success was statistically significant. The results of the Wilcoxon Test (fully tabulated in Appendix D, Table D.1) is graphically illustrated in Figure 7.25, where it is shown that the observed median is significantly different from the hypothesised median thereby indicating that there was significant agreement among software professionals that organisations have implemented organisational factors that are important for DevOps success.

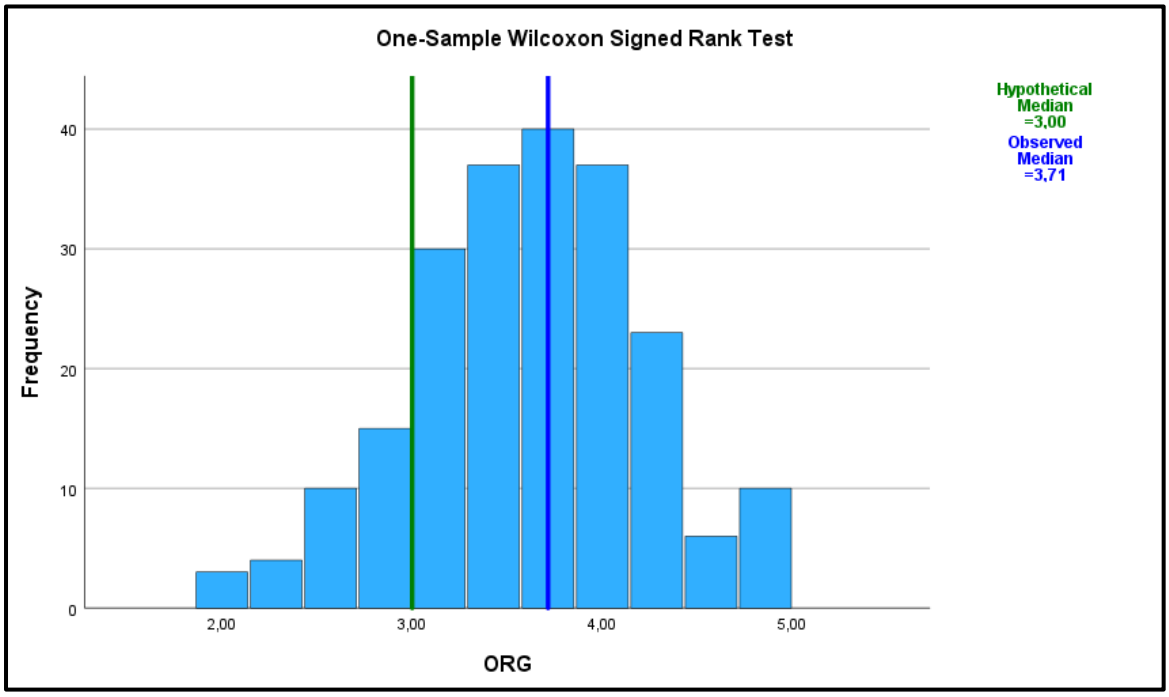

Figure 7.25: Central Tendency Data for Organisational Construct

People

The people construct sought to establish whether DevOps team members have the necessary characteristics (such as skills, knowledge, and willingness to learn) for DevOps. Figure 7.26 represents the software professionals' level of agreement with different items that made up the people construct. Over 70% of the respondents agreed or strongly agreed with five of the seven items measured. DevOps team members are willing to learn and innovate and DevOps team members are highly motivated were the two items that had the highest positive agreement with 89.3% and 81.3% of the respondents agreeing or strongly agreeing. Furthermore, 80% of the respondents agreed or strongly agreed that team members have a high level of technical competence and expertise and 78% agreed or strongly agreed that DevOps project management personnel adopted a management style

that encourages flexibility and creativity. However, only 69,8% agreed or strongly agreed that DevOps teams work in a cohesive, self-organising manner and 39.6% agreed or strongly agreed that management personnel have a positive relationship with the customer/business unit.

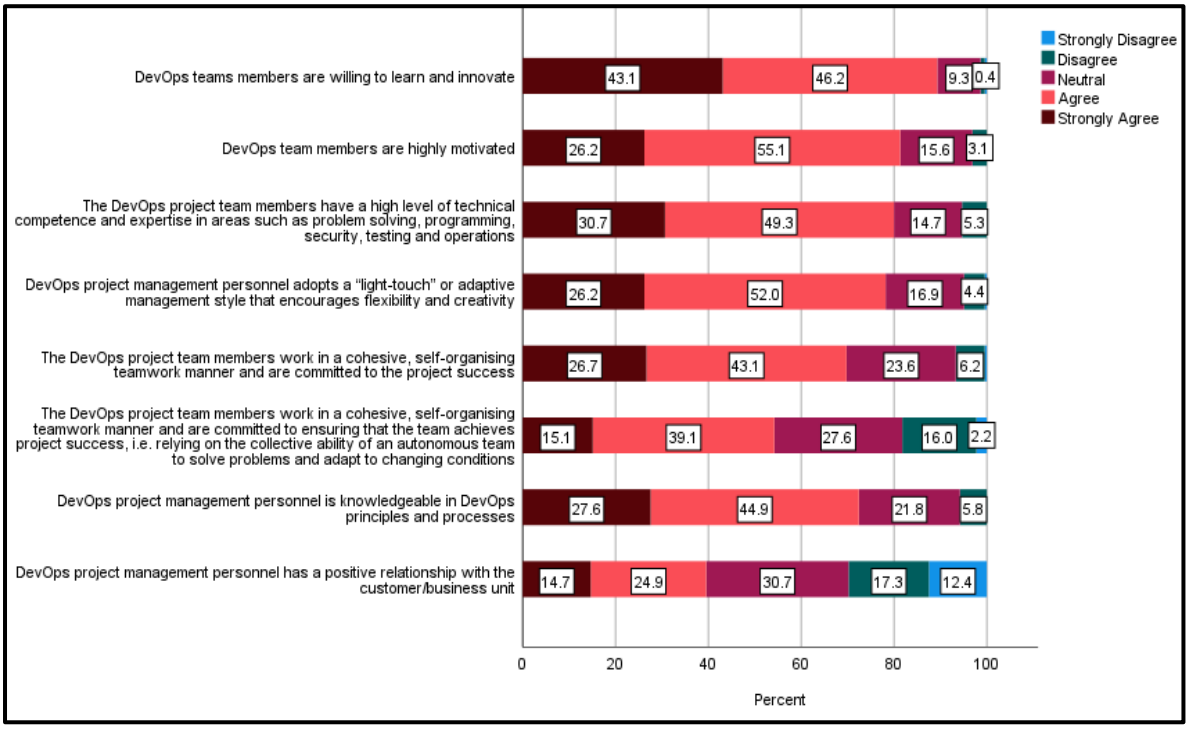

Figure 7.26: Frequency of the People Construct

For the People construct, the mean response was recorded at 3.86, the standard deviation (SD) =0.49, and the median = 3.88 (*M*=3.86; *SD* = 0.49; *Mdn*= 3.88).

The conflated view, presented in Figure 7.27 shows the majority (94.4%, n=215) positively perceive the People construct. This further indicates that most respondents agreed or strongly agreed that DevOps teams have the appropriate people factors to enable the success of the DevOps project.

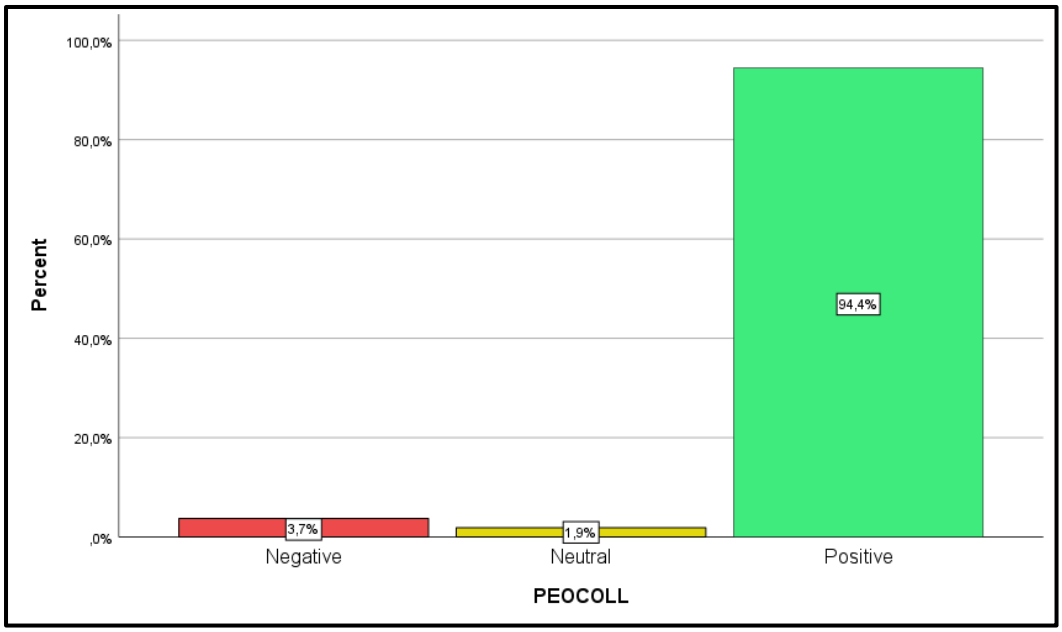

Figure 7.27: Sentiment Analysis of the People Construct

The Wilcoxon signed test was used to test if the agreement among respondents were significant. This test determined if the hypothetical median differed significantly from the observed median (Figure 7.28). The results of the Wilcoxon Test (Appendix D, Table D.1) showed a significant agreement among software professionals that DevOps teams have the necessary people factors such as motivation, habit and competence that are important for DevOps success.

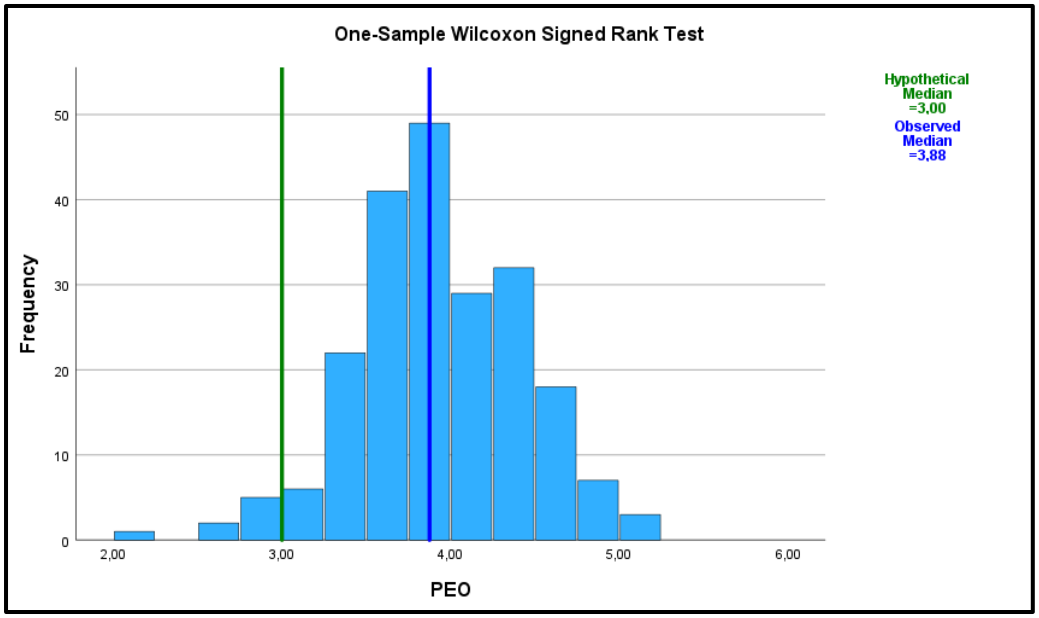

Figure 7.28: Central Tendency Data for People Construct

Project

For DevOps projects to be successful, specific project characteristics must be met. The project factor sought to determine whether these characteristics were part of the DevOps project. Figure 7.29 represents the software professionals' level of agreement with different items that made up the project

construct. Sixty-seven percent of the respondents agreed or strongly agreed that DevOps projects are characterised as mission-critical business software, 64% agreed or strongly agreed that DevOps projects have up-front risk analysis done, and 73.7% agreed strongly agreed that DevOps projects have a variable scope that leads to new system requirements. Also, 69.4% agreed or strongly agreed that DevOps projects have a dynamic, accelerated schedule. However, only 48.9% agreed or strongly agreed that detailed, upfront cost evaluations characterise DevOps projects.

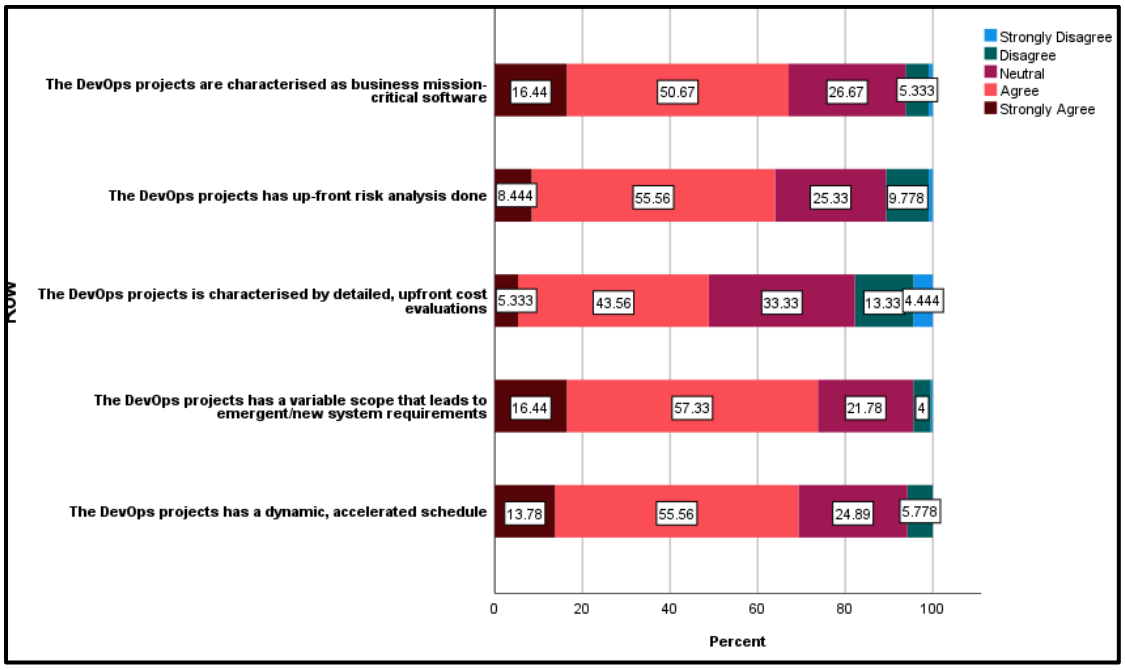

Figure 7.29: Frequency of the Project Construct

For the Project construct, the mean response was recorded at 3.69, the standard deviation (SD) =0.56, and the median = 3.80 (*M*=3.69; *SD* = 0.56; *Mdn*= 3.80).

The conflated view, presented in Figure 7.30 shows the majority (81.9%, n=215) of the respondents positively perceive the project construct.

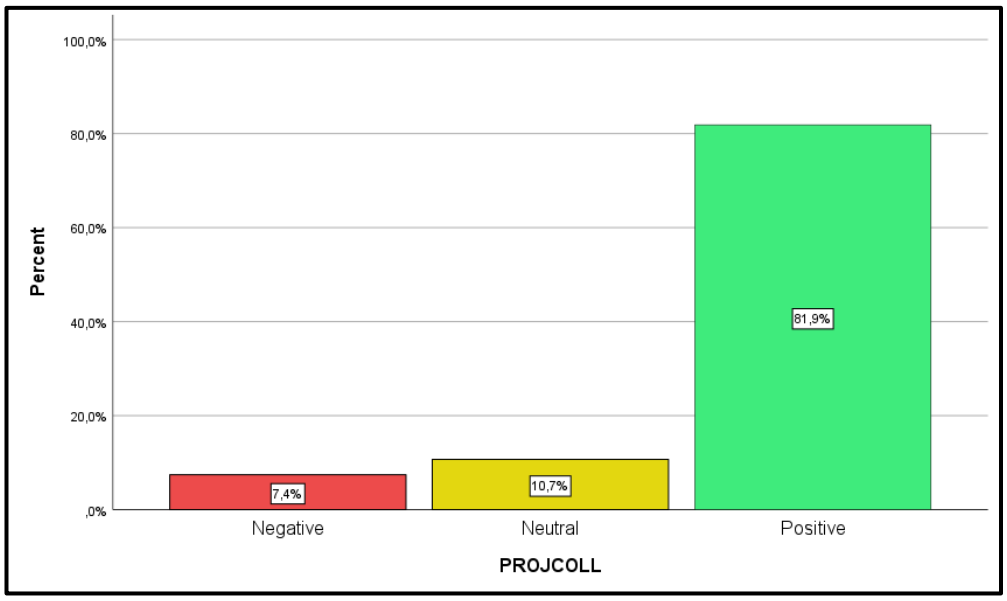

Figure 7.30: Sentiment Analysis of the Project Construct

The results from Figure 7.30 further indicates that most respondents agreed or strongly agreed that the DevOps project characteristics contributed to overall DevOps success. The Wilcoxon signed test was used to test if the agreement among respondents were significant. This test determined if the hypothetical median differed significantly from the observed median (Figure 7.31). The results of the Wilcoxon Test (Appendix D, Table D.1) showed a significant agreement among software professionals that DevOps teams have the necessary people factors such as motivation, habit and competence that are important for DevOps success.

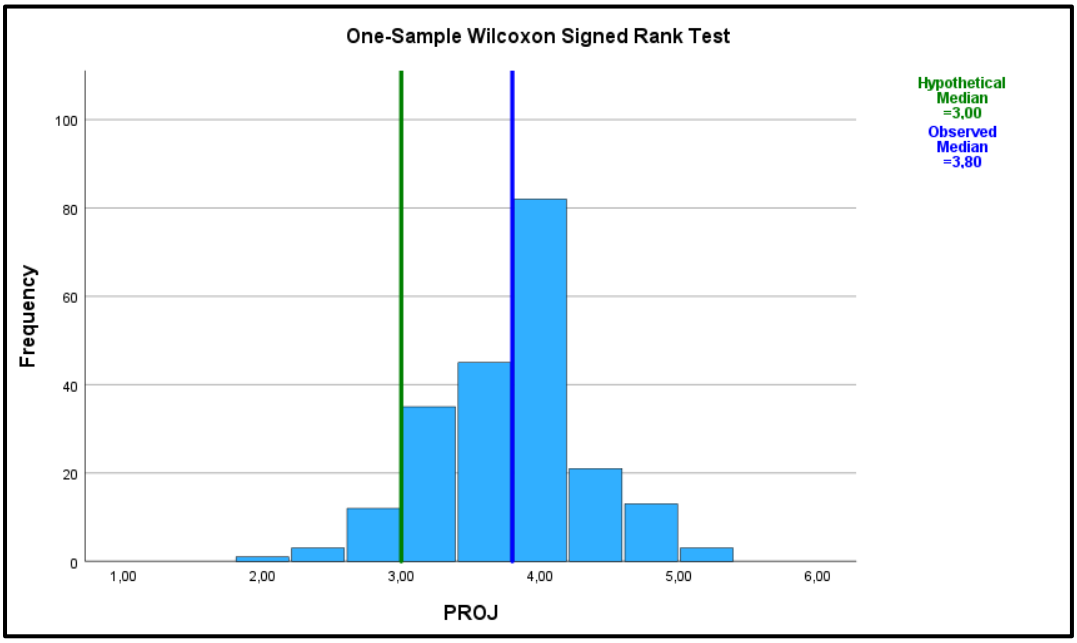

Figure 7.31: Central Tendency Data for Project Construct

Process

For DevOps projects to be successful, specific processes must be implemented. The process factor sought to determine whether these processes were part of the DevOps projects. Figure 7.32 represents the software professionals' level of agreement with different items that made up the process construct. The majority of the respondents (85.8%) agree or strongly agreed that DevOps projects continuously integrated team members' work, thereby reducing risk and identifying issues early in the software development life cycle, 84.9% of the respondents agreed or strongly agreed that the DevOps projects follow the practice of continuous delivery ensuring that the software can be released any time. Also, 79.5% agreed or strongly agreed that DevOps projects follow the practice of continuous testing, 74.2% agreed or strongly agreed that the DevOps projects were monitored continuously for early warning operation and quality defects that may occur in production, and 73.7% agreed or strongly agreed that DevOps projects follow a planning and management process.

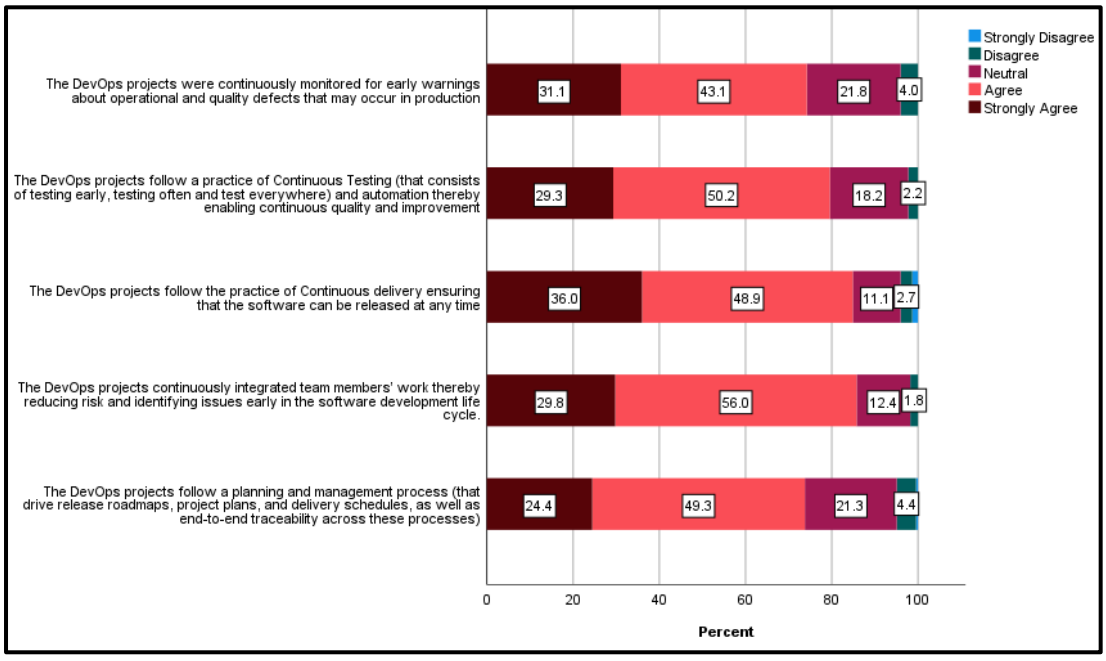

Figure 7.32: Frequency of the Process Construct

For the Process construct, the mean was recorded as 4.08, the standard deviation (SD) =0.58, and the median = 4.00 (*M*=4.08; *SD* = 0.58; *Mdn*= 4.0).

The conflated view, presented in Figure 7.32 shows the majority (94%, n=215) respondents positively perceive the process construct.

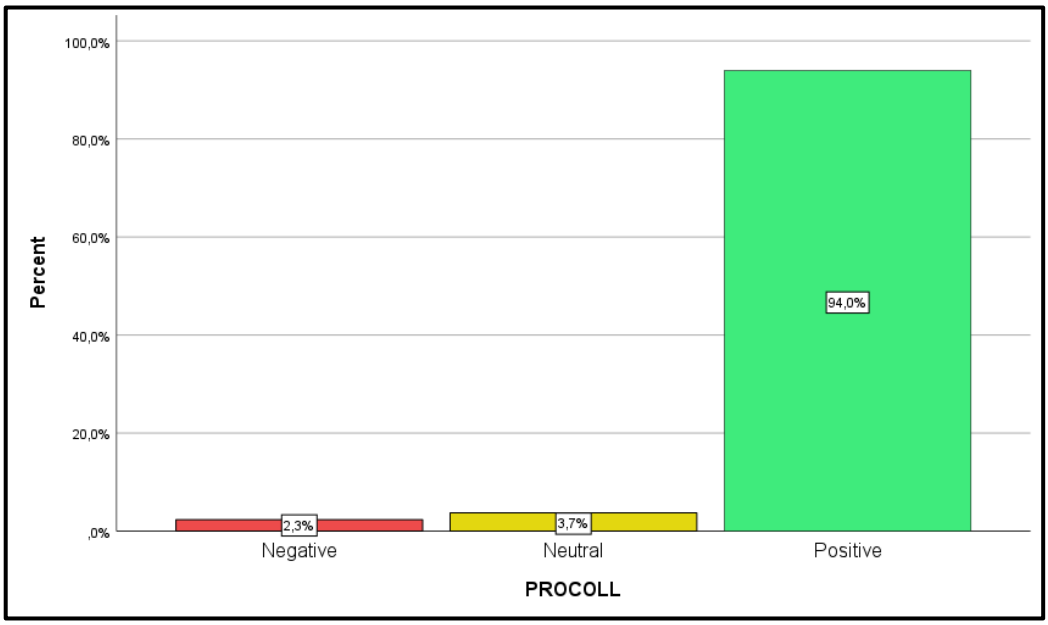

Figure 7.33: Sentiment Analysis of the Process Construct

The results from Figure 7.33 indicate that most respondents agreed or strongly agreed that DevOps projects implemented various processes to ensure their success. The Wilcoxon signed test was used to test if the agreement among respondents were significant. This test determined if the hypothetical median differed significantly from the observed median (Figure 7.34). The results of the Wilcoxon Test (Appendix D, Table D.1) showed a significant agreement among software professionals that DevOps projects follow processes for continuous testing, monitoring, and integration and follow a planning and management process.

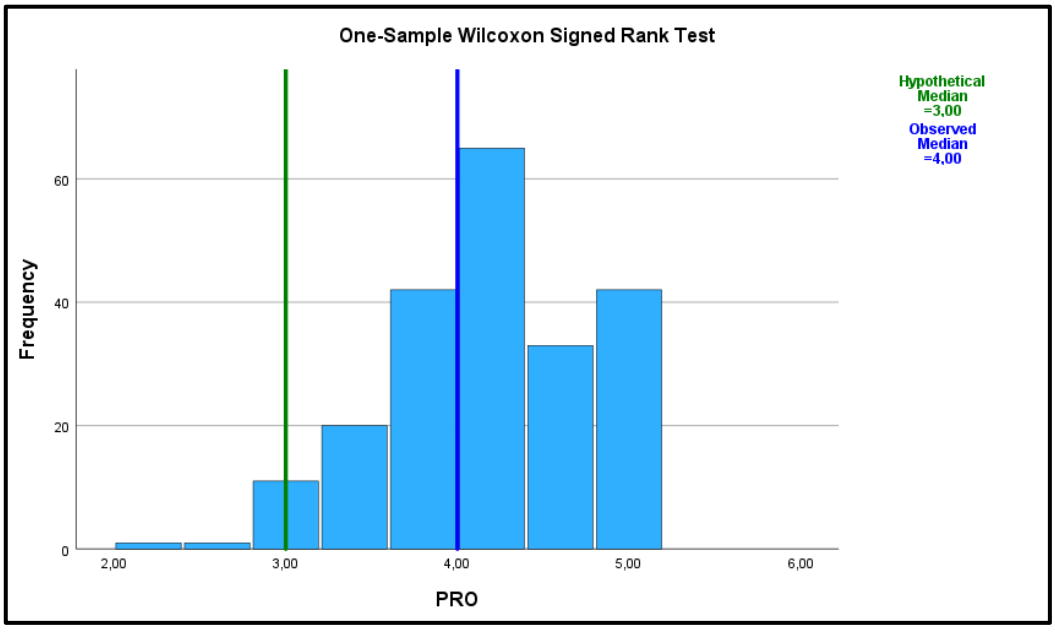

Figure 7.34: Histogram and Central Tendency Data for Process Construct

Culture

The team must exhibit certain behaviour and beliefs for DevOps projects to succeed. The culture factor sought to determine whether the behaviour and beliefs were part of the DevOps project team. Figure 7.35 represents the software professionals' level of agreement with different items that made up the culture construct. The majority of the respondents (86.7%) agree or strongly agreed that DevOps teams have a culture of collaboration, 85.7% of the respondents agreed or strongly agreed that the DevOps team have a culture of automating almost everything, 82.7% agreed or strongly agreed that DevOps team have a strong focus on continuous improvements to optimise performance, cost and speed of delivery. 80.5% agreed or strongly agreed that the DevOps team has a culture that espouses a shared understanding between key role players, and 75.2% agreed or strongly agreed that the DevOps team has a culture of embracing failures to turn failures into learning opportunities.

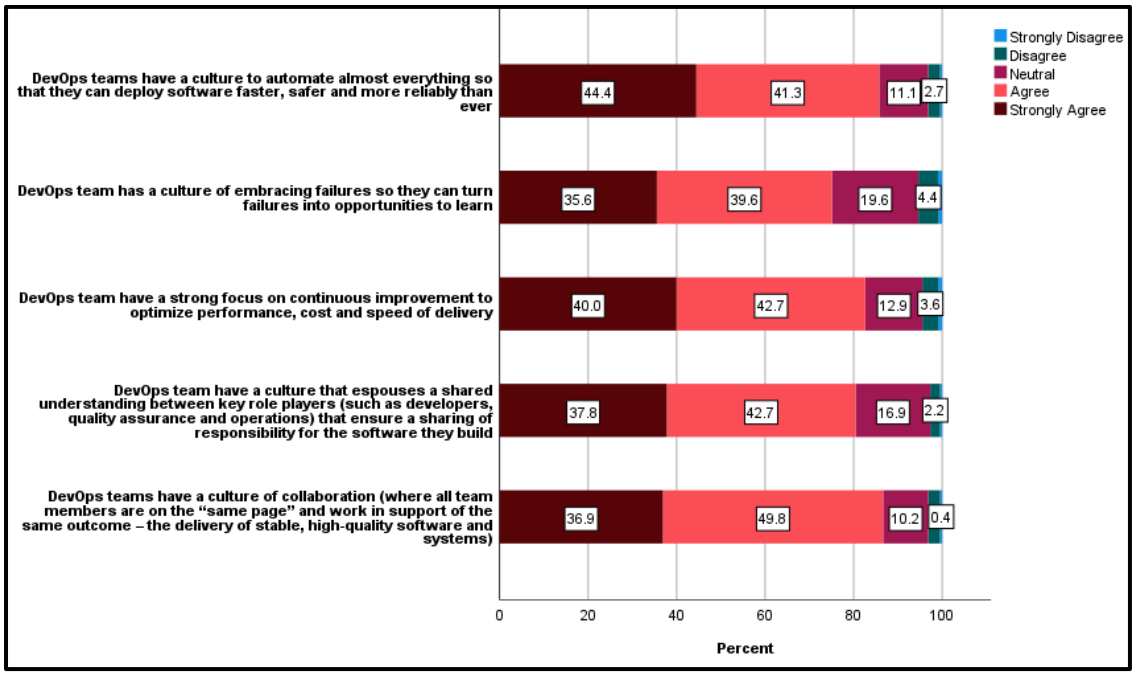

Figure 7.35: Frequency of the Culture Construct

For the construct of Culture, the mean was recorded as 4.19, the standard deviation (SD) =0.69, and the median = 4.00 (*M*=4.08; *SD* = 0.58; *Mdn*= 4.0).

Figure 7.36 illustrates the responses to negative, neutral and positive perceptions of the process construct.

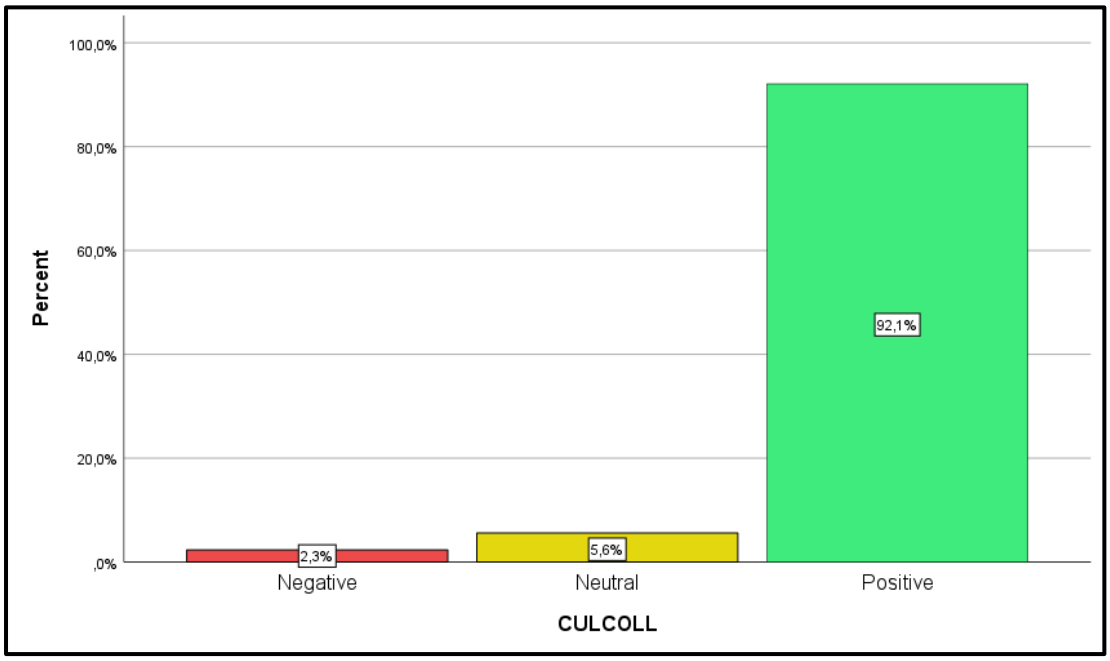

Figure 7.36: Sentiment Analysis of the Culture Construct

As shown in Figure 7.36, most respondents (92.1%, n=215) perceive the culture construct positively. This further indicates that most respondents agreed or strongly agreed that DevOps teams have a culture of embracing failures, automating everything, and a culture that espouses a shared understanding between crucial role players.

The Wilcoxon signed test was used to test if the agreement among respondents were significant. This test determined if the hypothetical median differed significantly from the observed median (Figure 7.37). The results of the Wilcoxon Test (Appendix D, Table D.1) showed a significant agreement.

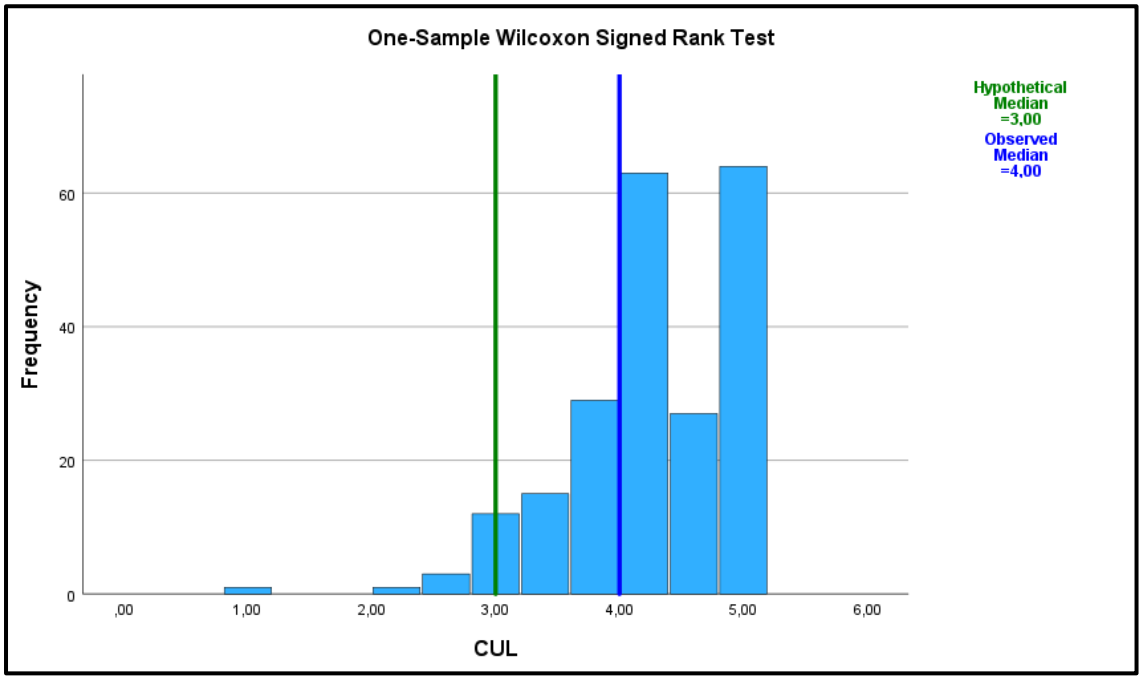

Figure 7.37: Central Tendency Data for the Culture Construct

Tools and Technologies

For DevOps projects to succeed, software professionals must use various technologies. The technology factor sought to determine the technologies used in DevOps projects. Figure 7.38 represents the software professionals' level of agreement with different items that made up the technology construct. The majority of the respondents (91.1%) agree or strongly agreed that DevOps projects use version control tools (such as Git and Github) to coordinate work among programmers and that DevOps projects use planning tools (such as Jira) for planning and project tracking allowing the team to ship code early and often. 90.1% agreed or strongly agreed that DevOps projects use continuous integration and delivery tools like Jenkins. 87.6% of the respondents agreed or strongly agreed that the DevOps projects use builds tools (such as Gradle or Maven) to automate builds, allowing for automatic download and configuration of dependencies or other libraries, 84.9% agreed or strongly agreed that DevOps projects use deployment tools such as docker to deliver software in packages called containers, and 83.6% agreed or strongly agreed that the DevOps projects use tools such as Kubernetes for container orchestration. Furthermore, 76.4% agreed or strongly agreed that DevOps projects use monitoring tools such as Splunk that provide real-time insight into errors across the infrastructure and 76.4% agreed or strongly agreed that DevOps projects use operational tools such as Chef that treats infrastructure as code.

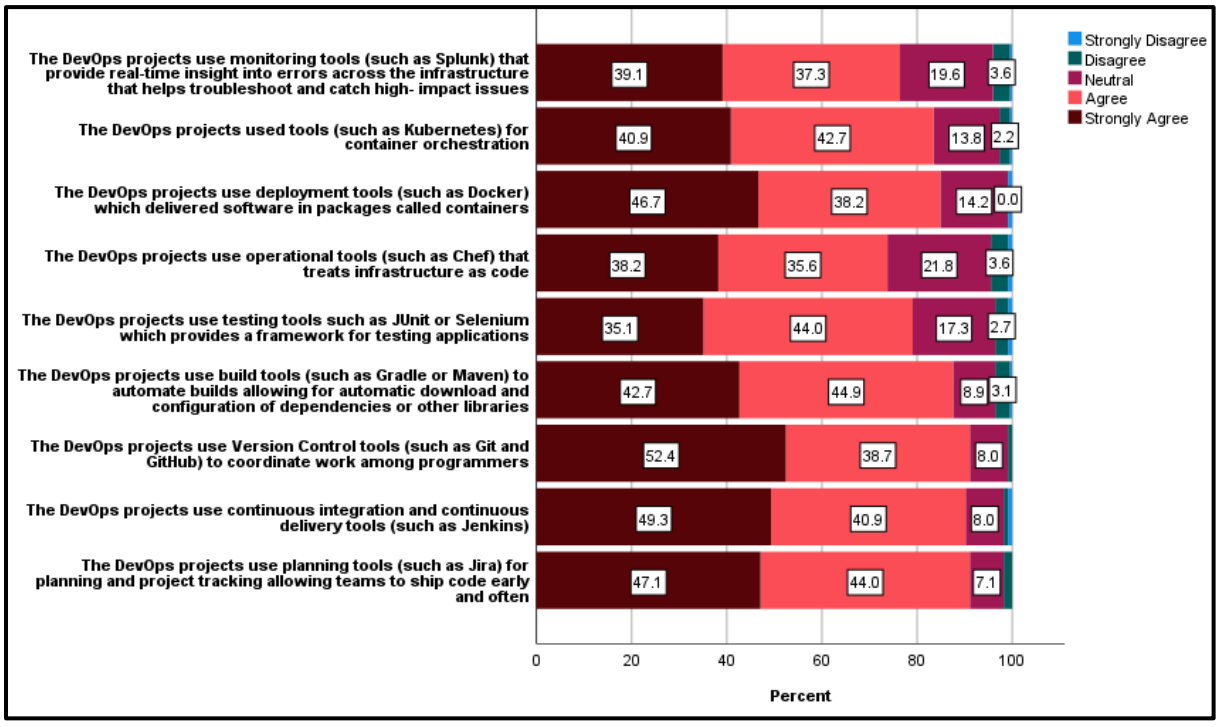

Figure 7.38: Frequency of the Technology Construct

For this construct of technology, the mean was reported to be 4.26, the standard deviation (SD) =0.61, and the median = 4.22 (*M*=4.26; *SD* = 0.61; *Mdn*= 4.22).

Figure 7.39 illustrates the responses to negative, neutral and positive perceptions of the technology construct. As shown in Figure 7.39, most respondents (92.9%, n=215) perceive the technology construct positively. This further indicates that most respondents agreed or strongly agreed that DevOps projects utilise various technologies for continuous delivery and deployment, monitoring and testing.

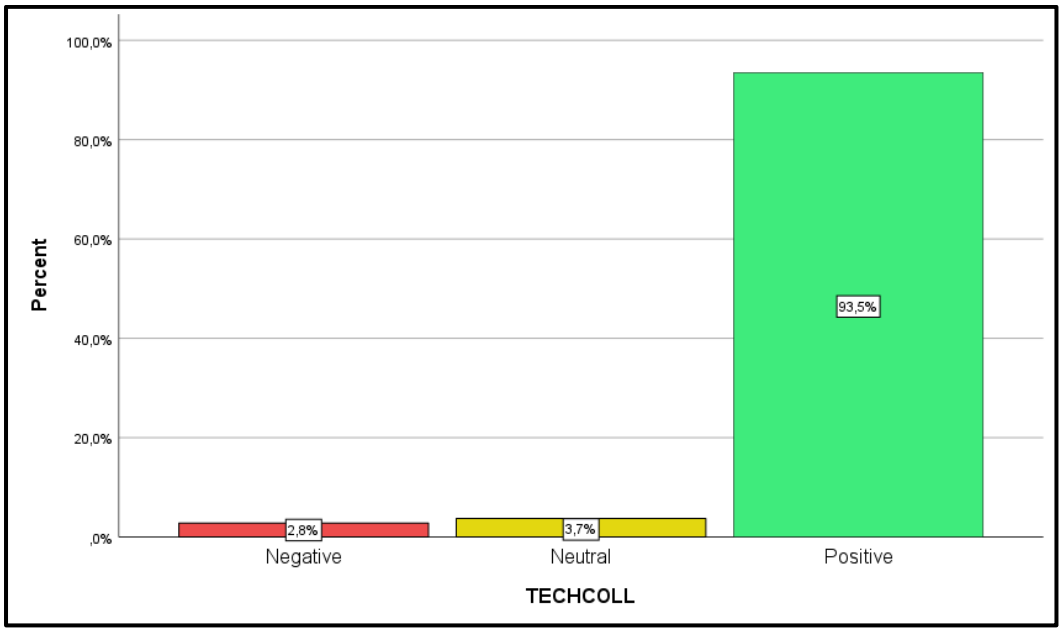

Figure 7.39: Sentiment Analysis of the Technology Construct

The Wilcoxon signed test was used to test if the agreement among respondents was significant. This test determined if the hypothetical median differed significantly from the observed median (Figure 7.40). The results of the Wilcoxon Test (Appendix D, Table D.1) showed a significant agreement.

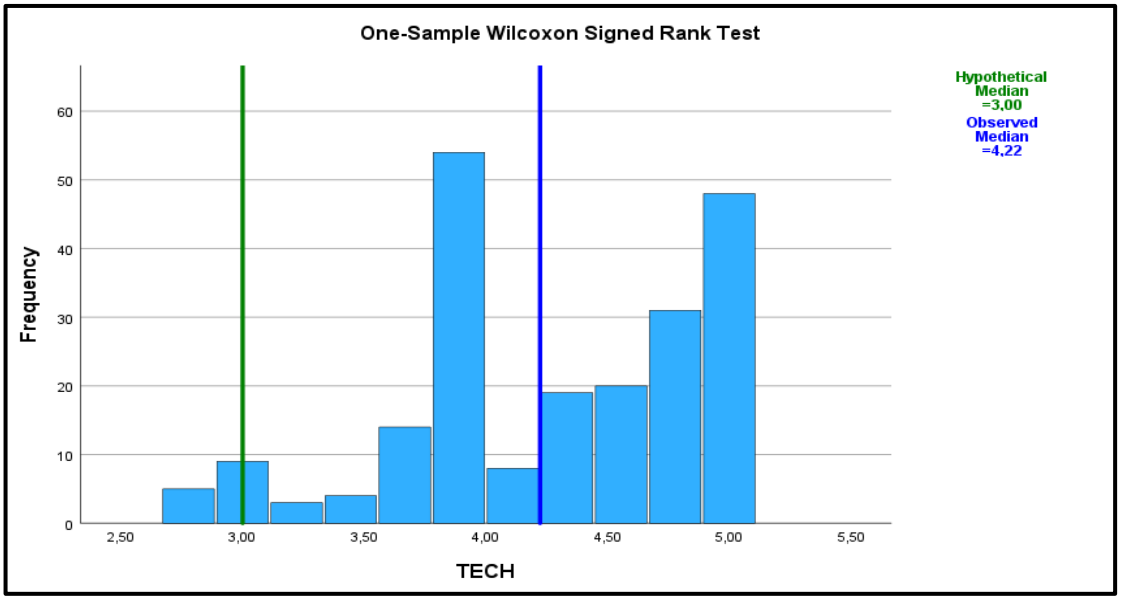

Figure 7.40: Central Tendency Data for the Technology Construct

Automation

For DevOps projects to succeed, software professionals must automate various software development tasks. The automation factor sought to determine the level of automation used in DevOps projects. Figure 7.40 represents the software professionals' level of agreement with different items that made up the automation construct. Most respondents (90.2%) agree or strongly agreed that DevOps projects use an automated build trigger for continuous integration. 84.4% agreed or strongly agreed that DevOps projects use automated execution for continuous testing, 80.5% of the respondents agreed or strongly agreed that the DevOps projects automatically provision for servers and installation of application code, 79.1% of the respondents agreed or strongly agreed that DevOps projects use an automated environment configuration for integration, UI and regression testing. Also, 78.6% agreed or strongly agreed that DevOps projects automatically provision for servers and installation of application code.

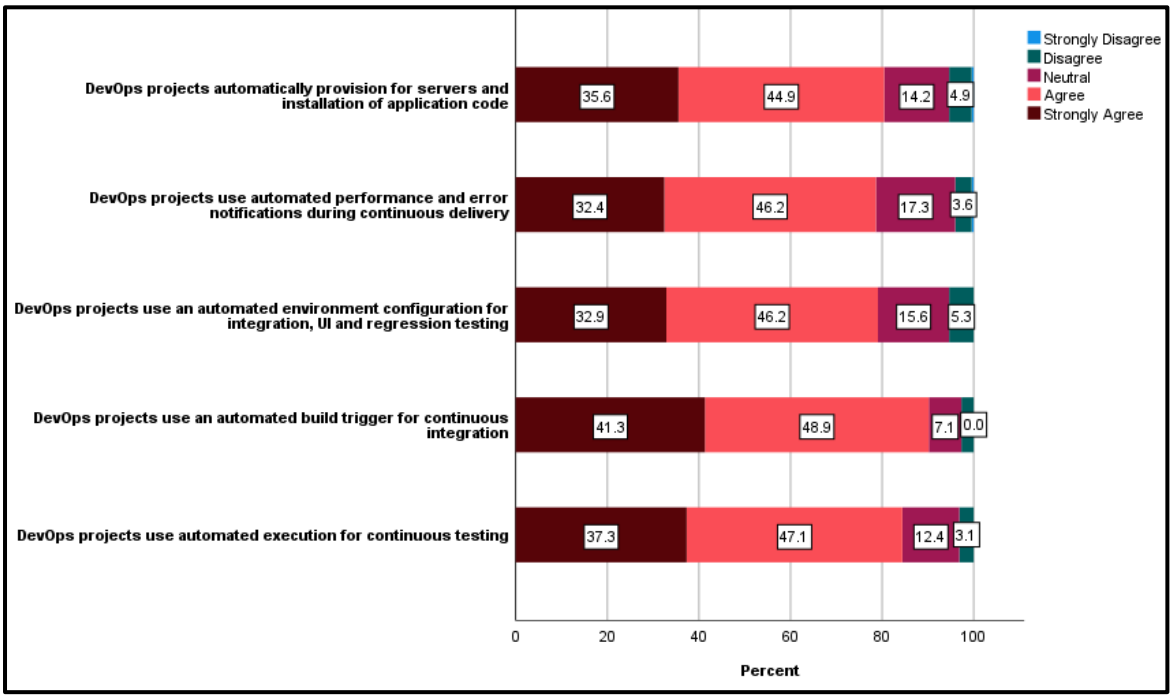

Figure 7.41: Frequency of the Automation Construct

For the construct of Automation, the mean was reported at 4.17, the standard deviation (SD) =0.64, and the median = 4.0 (*M*=4.17; *SD* = 0.64; *Mdn*= 4.0).

Figure 7.42 illustrates the responses to negative, neutral and positive perceptions of the automation construct. As shown in Figure 7.42, most respondents (94%, n=215) perceive the automation construct positively. This further indicates that most respondents agreed or strongly agreed that DevOps projects utilise automation for the provision of servers, performance and error notifications,

integration and regression testing, build triggers for continuous integration and execution for continuous testing.

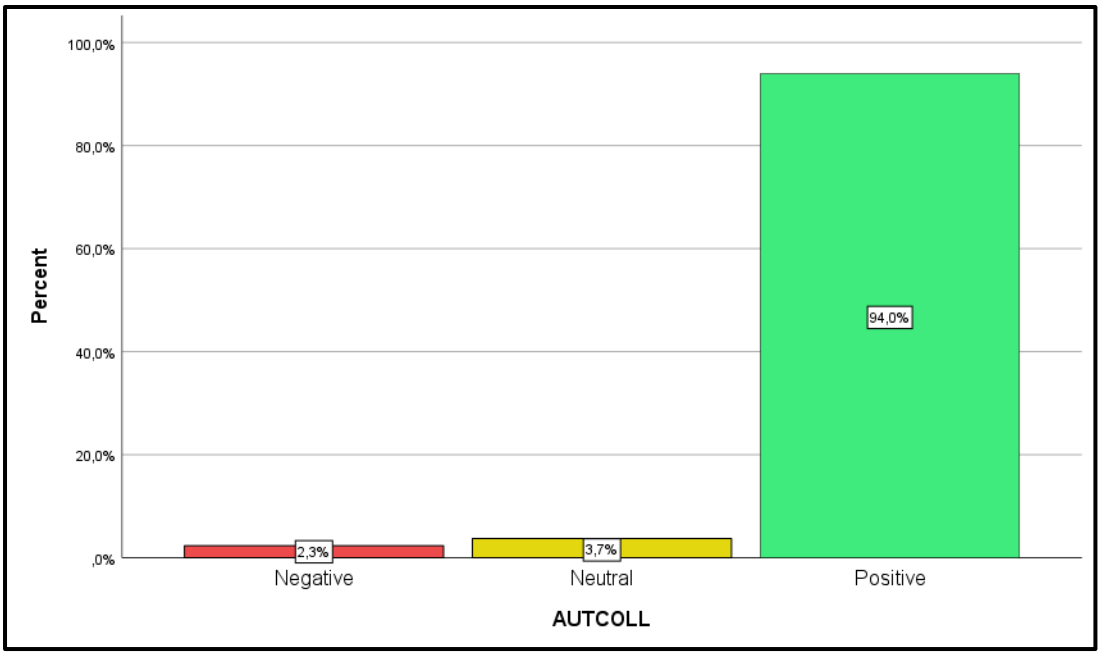

Figure 7.42: Sentiment Analysis of the Automation Construct

The Wilcoxon signed test was used to test if the agreement among respondents were significant. This test determined if the hypothetical median differed significantly from the observed median (Figure 7.43). The results of the Wilcoxon Test (Appendix D, Table D.1) showed a significant agreement.

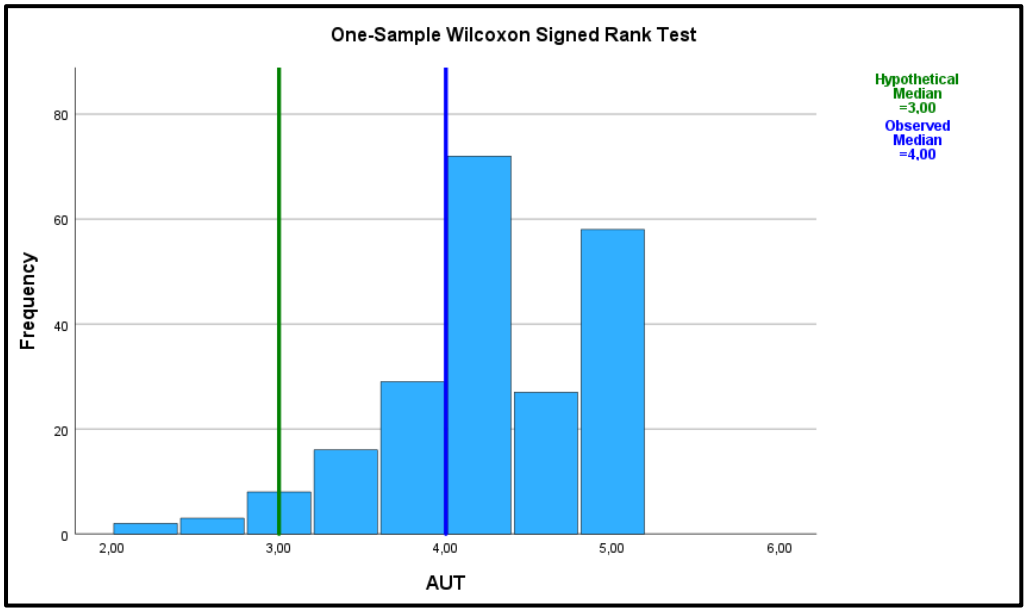

Figure 7.43: Central Tendency Data for the Automation Construct

Security

For DevOps projects to succeed, software professionals must implement security practices into the DevOps process. The security factor sought to determine the level of security implemented in DevOps projects. Figure 7.44 represents the software professionals' level of agreement with different items that made up the security construct. The majority of the respondents (79.6%) agree or strongly agreed that DevOps projects use code analysis for vulnerability testing and quality assurance, 76.5% agreed or strongly agreed that DevOps projects incorporate security into the full development lifecycle thereby not sacrificing necessary security measures in the pursuit Of CI/CD speed, 75.6 % agreed or strongly agreed that DevOps projects use a change management process to ensure that any developer can suggest a mission-critical security change, 75.5% agreed or strongly agreed that DevOps projects compared new features and updated against evolving threats and 70.6% of the respondents agreed or strongly agreed that the DevOps projects conduct periodic scans and do code reviews and penetration tests even after code is released.

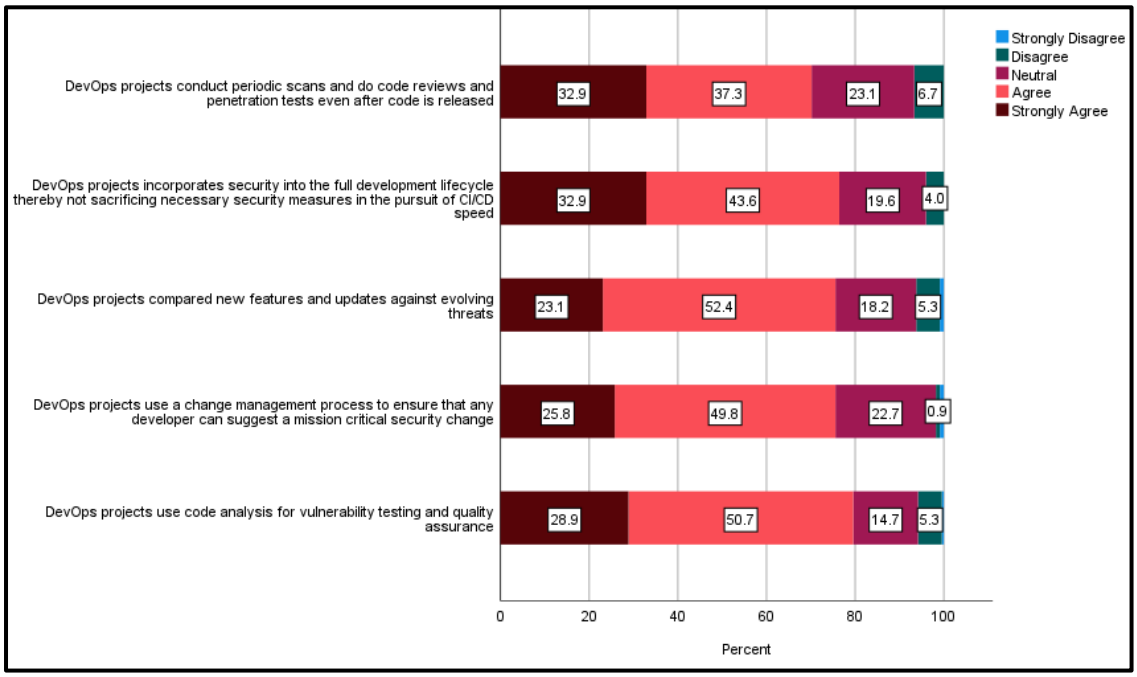

Figure 7.44: Frequency of the Security Construct

For this construct, the mean was 4.02, the standard deviation (SD) =0.64, and the median = 4.0 (*M*=4.02; *SD* = 0.64; *Mdn*= 4.0).

Figure 7.45 illustrates the responses to negative, neutral and positive perceptions of the automation construct. As shown in Figure 7.45, most respondents (90.2%, n=215) positively perceive the security construct. This further indicates that most respondents agreed or strongly agreed that DevOps projects incorporate security into the DevOps process.

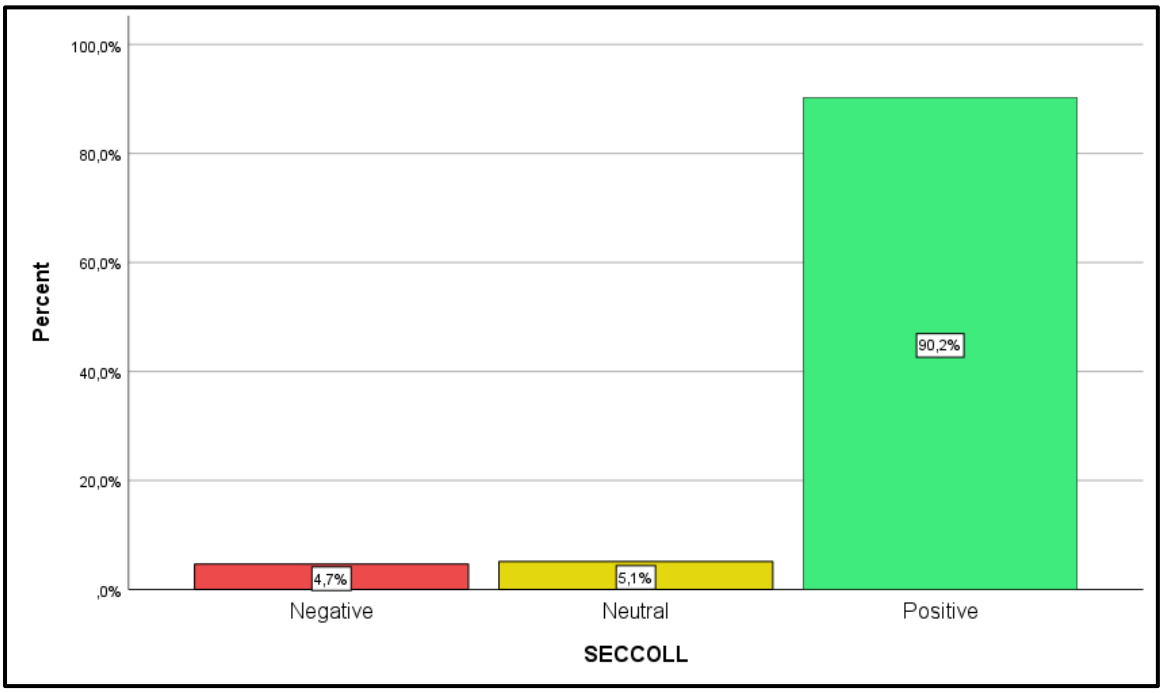

Figure 7.45: Sentiment Analysis of the Security Construct

The Wilcoxon signed test was used to test if the agreement among respondents was significant. This test determined if the hypothetical median differed significantly from the observed median (Figure 7.46). The results of the Wilcoxon Test (Appendix D, Table D.1) showed a significant agreement.

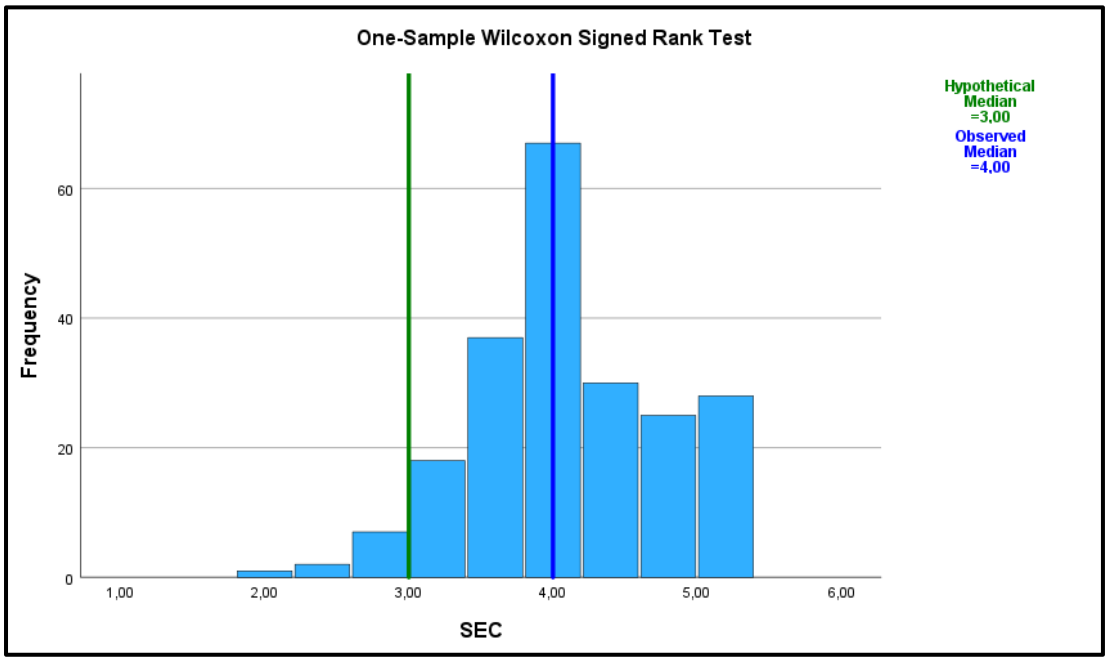

Figure 7.46: Central Tendency Data for the Security Construct

Cloud

DevOps projects developed and implemented via the cloud platform can increase the success of DevOps projects. The cloud factor sought to determine the level at which the cloud platform is used in DevOps projects. Figure 7.47 represents the software professionals' level of agreement with different items that made up the cloud construct. The majority of the respondents (74.7%) agree or strongly agreed that DevOps projects use Infrastructure as Code for the provisioning of servers and installation of application code, 71.5% agreed or strongly agreed that DevOps projects use cloud platforms to provide advanced automation, 65.3% agreed or strongly agreed that DevOps projects use a centralised cloud platform for testing, deployment, monitoring and operating the production workloads, 63.1% agreed or strongly agreed that DevOps projects use provisioned multiple cloud-based test environments to enhance quality and productivity during testing.

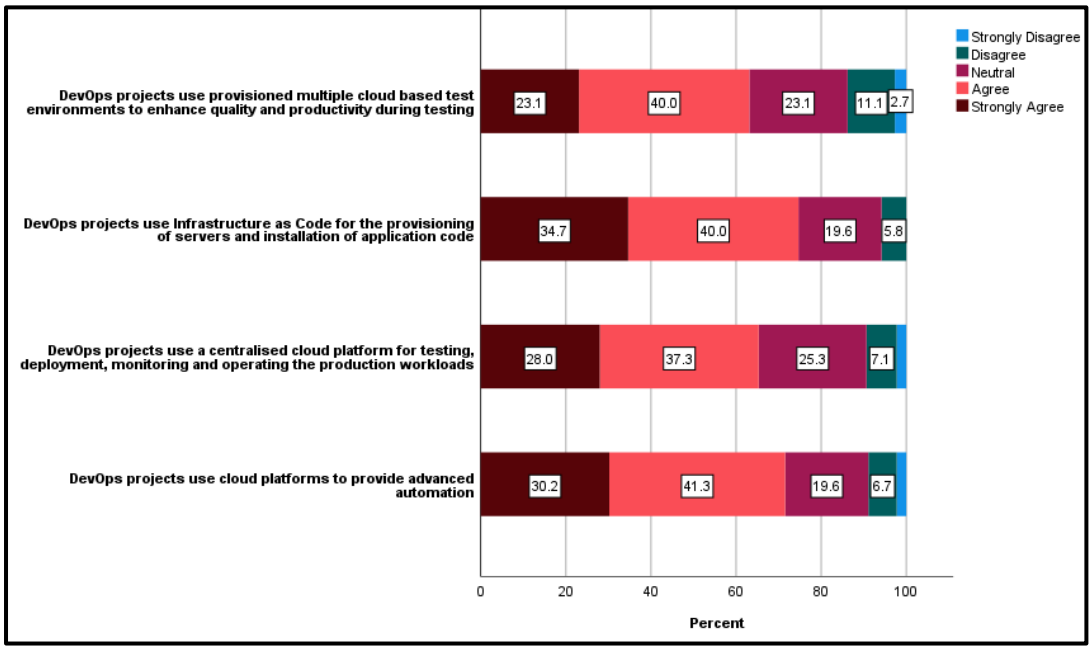

Figure 7.47: Frequency of the Cloud Construct

For Cloud construct, the mean was 3.87, the standard deviation (SD) =0.81, and the median = 4.0 (*M*=3.87; *SD* = 0.81; *Mdn*= 4.0).

Figure 7.48 illustrates the responses to negative, neutral and positive perceptions of the Cloud construct. This further indicates that most respondents agreed or strongly agreed that DevOps projects incorporate cloud technologies into the DevOps process.

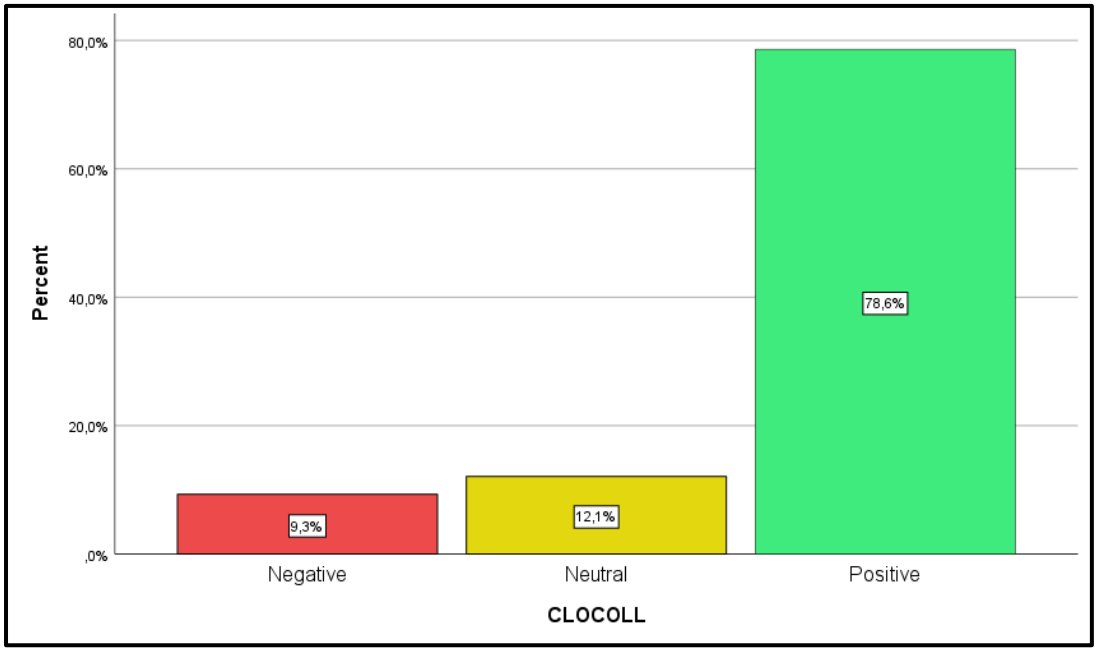

Figure 7.48: Sentiment Analysis of the Cloud Construct

The Wilcoxon signed test was used to test if the agreement among respondents was significant. This test determined if the hypothetical median differed significantly from the observed median (Figure 7.49). The results of the Wilcoxon Test (Appendix D, Table D.1) showed a significant agreement.

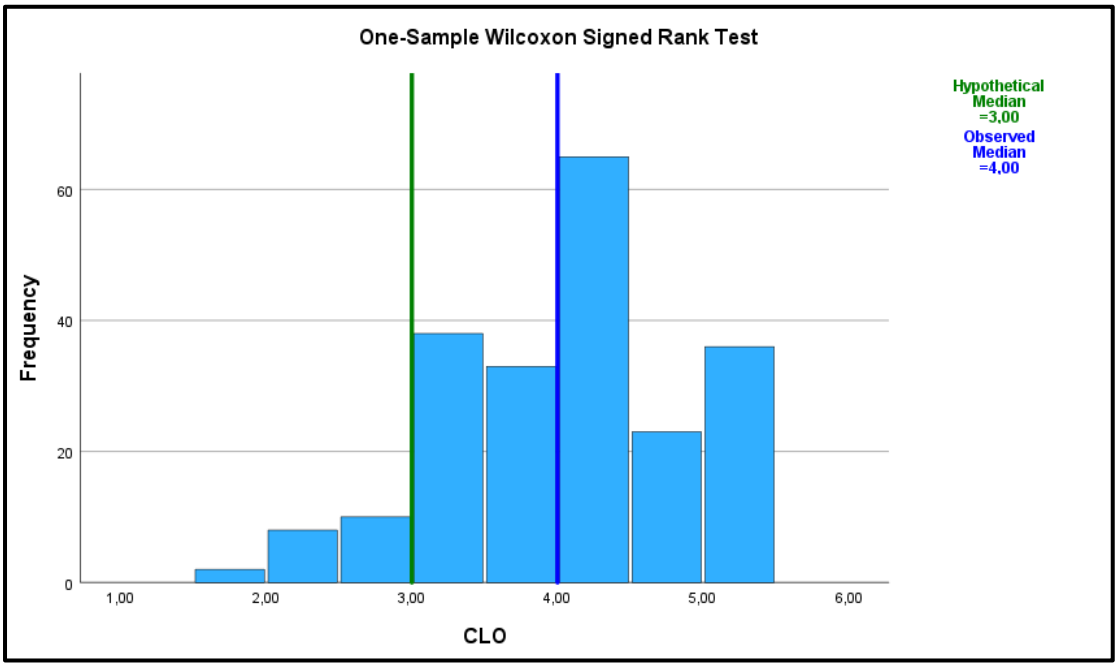

Figure 7.49: Central Tendency Data for the Cloud Construct

Actual Success

The actual success DevOps projects can be measured based on different criteria. Based on these criteria, the actual success factor sought to determine the level of success of DevOps projects. Figure 7.50 represents the software professionals' level of agreement with different items that made up the actual success construct. The majority of the respondents (87.5%) agree or strongly agreed that DevOps projects are successful in terms of deployment success rate, 84.4% agreed or strongly agreed that DevOps projects are successful in terms of meeting the customer/end user expectations of the system and in terms of quality of the project outcome or of the resulting software project, 84% agreed or strongly agreed that DevOps projects are successful in terms of the speed of deployments, 81.7% agreed or strongly agreed that DevOps projects are successful in terms of frequency of deployments and 76.4% are successful in terms of speed of build verification.

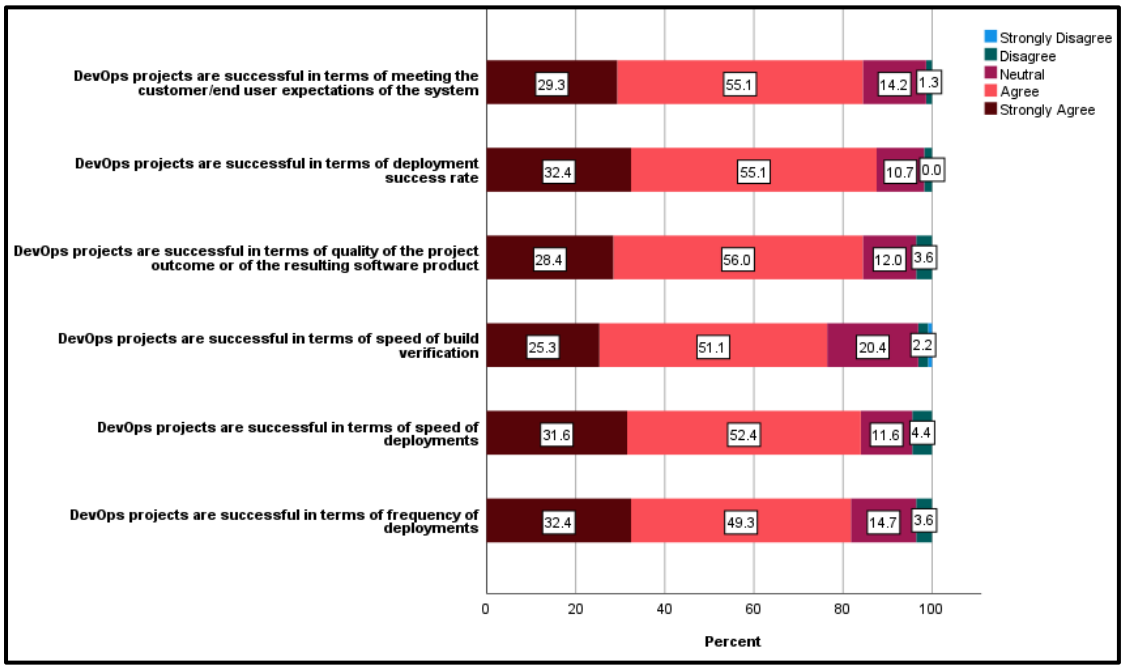

Figure 7.50: Frequency of the Actual Success Construct

For the construct of Actual Success, the mean was recorded as 4.1, the standard deviation (SD) =0.60, and the median = 4.11 (*M*=4.1; *SD* = 0.6; *Mdn*= 4.11).

Figure 7.51 illustrates the responses to negative, neutral and positive perceptions of the automation construct. As shown in Figure 51, the majority (91.6%, n=215) of the respondents have a positive perception of the cloud construct. This further indicates that most respondents agreed or strongly agreed that DevOps projects were successful.

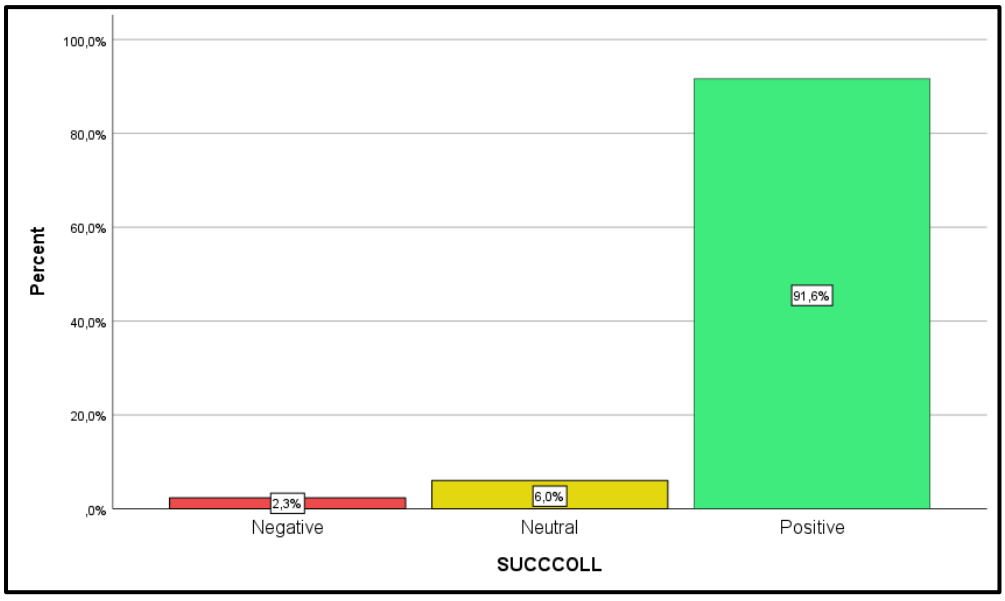

Figure 7.51: Sentiment Analysis of the Actual Success Construct

To test if the positive perception towards Actual Success is statistically significant, the Wilcoxon signed test was performed to test if the hypothetical median was significantly different from the observed median (Figure 7.52). The results of the Wilcoxon Test (Appendix D, Table D.1) showed a significant agreement.

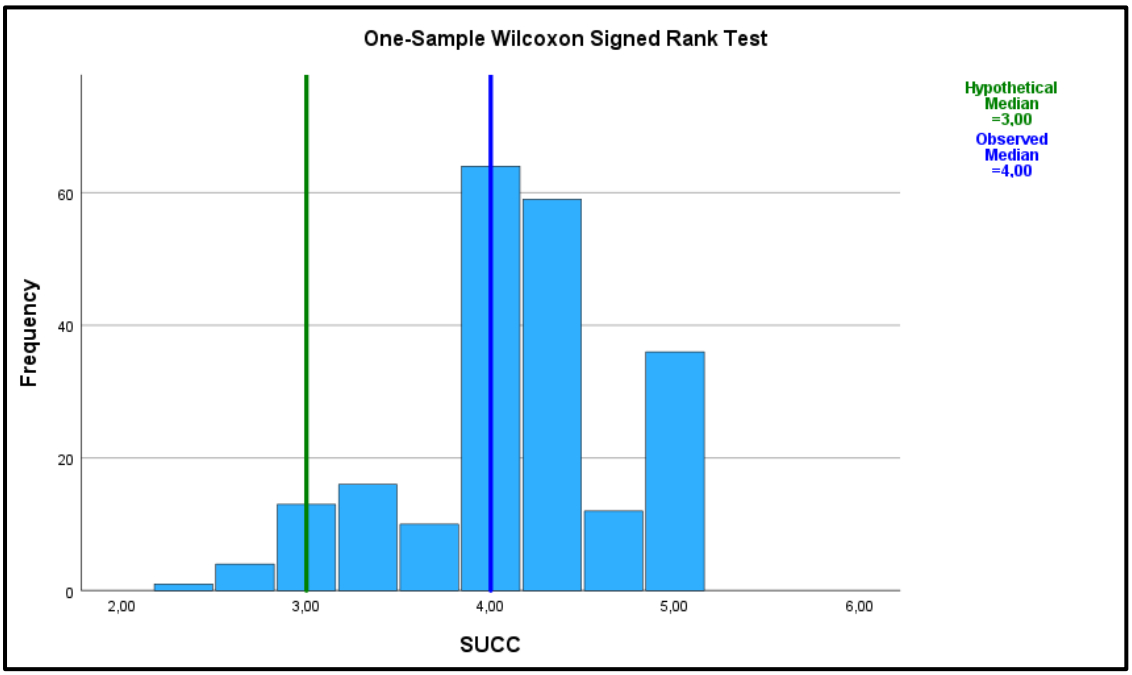

Figure 7.52: Central Tendency Data for Actual Success Construct

The results from Figure 7.52 suggest that DevOps practitioners have indicated that DevOps use contributes positively towards actual project success.

7.3 INFERENTIAL STATISTICS

7.3.1 Exploratory Factor Analysis (EFA)

An EFA was performed using principal axis factoring (PAF) and Promax rotation. The main advantages of PAF are that it does not require the normal distribution of data (Fabrigar et al., 1999) that is commonly found in Likert scale data and is more suited for small sample sizes with few indicators per factor (de Winter & Dodou, 2012).

There are two types of rotation methods: oblique and orthogonal. Oblique rotation is based on the correlation of variables, whereas orthogonal assumes that variables are uncorrelated. Field (2005) states that orthogonal rotation is not a realistic portrayal of data in real life since variables are expected to be correlated and, therefore, are not a good option for EFA. Osborne (2015, p. 5) states that "In the social sciences (and many other sciences, such as biomedical sciences), we generally

expect some correlation among factors since behaviour is rarely partitioned into neatly packaged units that function independently of one another. Therefore, using orthogonal rotation potentially results in a less useful solution where factors are correlated". Researchers have thus recommended oblique rotation to enable the intercorrelation of factors (Flora et al., 2012; Gaskin & Happell, 2014; Norris & Lecavalier, 2009). Oblique rotations can be implemented using direct Oblimin or Promax, with Promax rotation having the advantage of being conceptually simple and fast and producing more replicable results (Abdi, 2003; Rennie, 1997).

Before executing EFA, two essential assumptions must be checked to determine if the data is appropriate for factor analysis. The Kaiser-Meyer-Olkin (KMO) Measure of Sampling Adequacy tests whether the sample size is adequate for the overall model and for each variable (Shrestha, 2021). KMO values ranging from 0.8 to 1.0 indicate adequate sampling. KMO values ranging from 0.7 to 0.79 are considered average, while values ranging from 0.6 to 0.69 are considered poor. KMO values less than 0.6 indicate that the sampling is insufficient and that corrective action should be taken. As indicated in Table 7.4, the KMO value was 0.86, indicating an adequate sample size. The other test is Bartlett's Test of Sphericity, which checks whether the correlation matrix is an identity matrix. An identity matrix indicates that the variables are uncorrelated, so the data will not be suitable for factor analysis. A significant value $p<0.05$ was obtained for Bartlett's Test of Sphericity, indicating that it is not an identity matrix and that the variables are correlated. Hence, based on Bartlett's test for Sphericity and the KMO, the data can be analysed further using factor analysis.

Table 7.4: KMO and Bartlett's Test of Sphericity

KMO and Bartlett's Test		
Kaiser-Meyer-Olkin Measure of Sampling Adequacy.		0.857
Bartlett's Test of Sphericity	Approx. Chi-Square	19804.947
	Df	3828
	Sig.	0

Based on the KMO and Bartlett's test results (Table 7.4), EFA was carried out by extracting factors with eigenvalues> 1 and suppressing loading less than 0.3 (Field, 2013). The screeplot (Figure 7.53) shows an initial extraction of 20 factors with a total variance of 72%. However, the pattern matrix indicated that some items were only loaded with a single factor and others with multiple factors. Several analysis cycles resulted in the people, facilitating conditions and habit constructs being removed. Furthermore, items, as documented in Table 7.5, were also deleted.

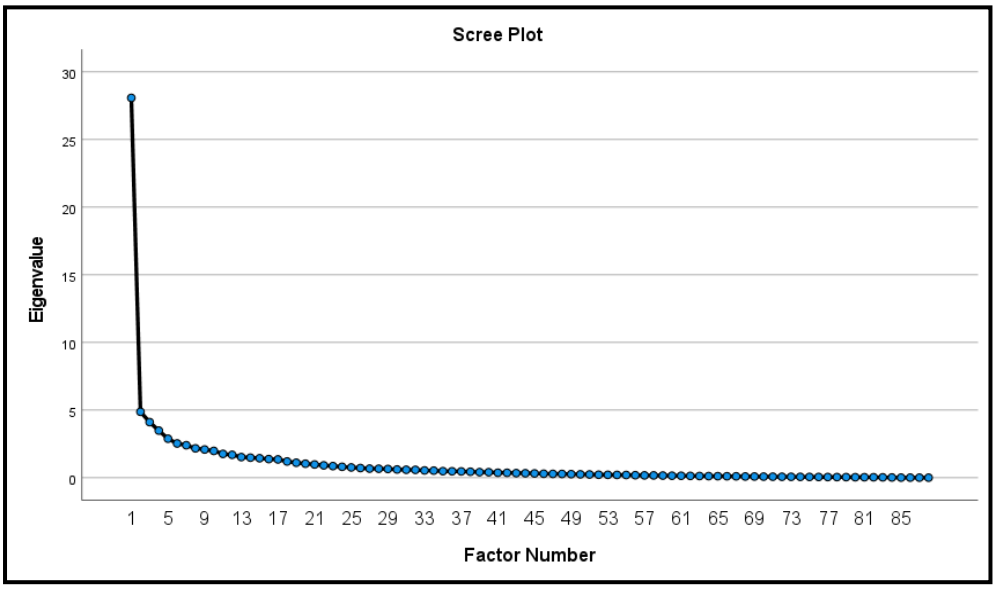

Figure 7.53: ScreePlot

Table 7.5: Items deleted after EFA

Construct	Item Name	Description
ORGANISATIONAL	ORG1	The DevOps project team is collocated, i.e. all team members worked in the same location for ease of communication and casual, and constant contact.
	ORG2	The organisation has a cooperative culture instead of hierarchal.
	ORG3	The organisation has an oral culture placing high value on face-to-face communication style.
	ORG5	The organisation has a reward system that acknowledges DevOps behaviour.
PROCESS	PRO1	The DevOps projects follow a planning and management process (that drive release roadmaps, project plans, and delivery schedules, as well as end-to-end traceability across these processes)
PROJECT	PROJ1	The DevOps projects has a dynamic, accelerated schedule.
	PROJ2	The DevOps projects has a variable scope that leads to emergent/new system requirements.
TECHNOLOGIES	TECH6	The DevOps projects use operational tools (such as Chef) that treats infrastructure as code.
	TECH8	The DevOps projects used tools (such as Kubernetes) for container orchestration.
CLOUD	CLO3	DevOps projects use Infrastructure as Code for the provisioning of servers and installation of application code.
SUCCESS	SUCC1	DevOps projects are successful in terms of frequency of deployments.

Once the 11 items listed in Table 7.5 was removed, it resulted in the extraction of 14 factors that accounted for a total variance of 69.3% for the study's data. Table 7.6 shows the final pattern matrix with the loadings of the different items on the 14 factors.

Table 7.6: Final Pattern Matrix

Pattern Matrix[a]		
Factor 1	ORG4	0.39
	ORG6	0.731
	ORG7	0.488
Factor 2	SEC1	0.765
	SEC2	0.63
	SEC3	0.523
	SEC4	0.583
	SEC5	0.77
Factor 3	PRO2	0.546
	PRO3	0.733
	PRO4	0.688
	PRO5	0.654
Factor 4	PROJ3	0.749
	PROJ4	0.762
	PROJ5	0.516
Factor 5	TECH1	0.731
	TECH2	0.861
	TECH3	0.873
	TECH4	0.861
	TECH5	0.581
	TECH7	0.6
	TECH9	0.606
Factor 6	AUT1	0.739
	AUT2	0.646
	AUT3	0.557
	AUT4	0.709
	AUT5	0.816
Factor 7	CLO1	0.879
	CLO2	0.871
	CLO4	0.768
Factor 8	CUL1	0.573
	CUL2	0.72
	CUL3	0.969
	CUL4	0.87
	CUL5	0.687

Factor 9	SUCC2	0.663
	SUCC3	0.716
	SUCC4	0.847
	SUCC5	0.648
	SUCC6	0.589
Factor 10	PER1	0.91
	PER2	0.846
	PER3	0.915
	PER4	0.862
Factor 11	EFF1	0.729
	EFF2	0.751
	EFF3	0.708
	EFF4	0.716
	EFF5	0.846
Factor 12	MOT1	0.57
	MOT2	0.882
	MOT3	0.611
	MOT4	0.879
Factor 13	SOC1	0.892
	SOC2	0.946
	SOC3	0.769
Factor 14	INT1	0.872
	INT2	0.805
	INT3	0.76
	INT4	0.685
Extraction Method: Principal Axis Factoring; Rotation Method: Promax with Kaiser Normalization.[a]; a. rotation converged in 9 iterations.		

7.3.2 Structural Equation Modelling (SEM)

Assumptions for SEM

According to Kline (2010), for SEM to succeed, certain basic assumptions are required, such as large sample sizes, no missing data, no univariate and multivariate normality, linearity in the data and multicollinearity. These assumptions are discussed in the context of the current study.

Missing data

As stated at the beginning of this chapter, there is no missing data since the survey was administered online, and every question had a required field. Also, all unengaged responses were deleted.

Testing the normality assumption

The statistical package AMOS was used to test for univariate and multivariate normality. The output from AMOS is commonly analysed by assessing the skewness and kurtosis. Lei and Lomax (2005, p. 2) categorised non-normality into three groups, stating that "The absolute values of skewness and kurtosis less than 1.0 as slight nonnormality, the values between 1.0 and about 2.3 as moderate

nonnormality, and the values beyond 2.3 as severe nonnormality". Greater clarity is provided by Kim (2013), who suggests that normality can be assessed by inspecting the Z-scores, also called critical ratios (CR). The CR must be adjusted based on sample size and suggest that the data is non-normal if the absolute value for CR is greater than 3.29 for a sample size between 50 - 300. Table 7.10 provides a snapshot of the output from AMOS. According to the Lei and Lomax (2005) classification of normality, some items display univariate moderate and severe non-normality (highlighted in yellow) since skewness and kurtosis values ranged from 1.016 to 3.435. Also, some items' critical ratios for skewness and kurtosis were above 3.29 (highlighted in red), indicating non-normality. Furthermore, according to Bentler (2005) multivariate normality is also violated since kurtosis's CR value is greater than 5 (highlighted in blue in Table 7.10). This finding is common in social science research since real-world data display univariate and multivariate non-normality (Gao et al., 2008; Kumar, 2015; Micceri, 1989).

Table 7.7: Univariate and multivariate normality

Variable	min	max	skew	c.r.	kurtosis	c.r.
CUL1	1,000	5,000	-,913	-5,463	1,366	4,089
PRO3	1,000	5,000	-1,159	-6,938	2,222	6,651
PRO4	2,000	5,000	-,397	-2,375	-,285	-,854
PRO5	2,000	5,000	-,451	-2,702	-,499	-1,494
TECH7	1,000	5,000	-1,059	-6,340	1,482	4,435
TECH5	1,000	5,000	-,854	-5,114	,841	2,516
TECH4	1,000	5,000	-1,092	-6,534	1,625	4,864
TECH3	2,000	5,000	-,969	-5,803	,510	1,527
TECH2	1,000	5,000	-1,415	-8,471	3,435	10,281
TECH1	2,000	5,000	-,994	-5,950	,999	2,991
CUL5	1,000	5,000	-1,016	-6,079	1,078	3,228
CUL4	1,000	5,000	-,768	-4,594	,276	,827
CUL3	1,000	5,000	-1,045	-6,254	1,243	3,720
AUT5	1,000	5,000	-,859	-5,141	,594	1,776
AUT3	2,000	5,000	-,657	-3,932	,018	,054
AUT2	2,000	5,000	-,844	-5,052	,943	2,822
AUT1	2,000	5,000	-,661	-3,957	,185	,553
SUCC6	2,000	5,000	-,266	-1,593	-,348	-1,042
PER4	2,000	5,000	-1,031	-6,172	1,001	2,997
PER3	2,000	5,000	-1,076	-6,439	,860	2,573
PER2	2,000	5,000	-,919	-5,500	,636	1,903
PER1	2,000	5,000	-1,109	-6,637	,745	2,230
Multivariate					679,692	57,772

Non-normality poses a problem when analysis is performed in AMOS using the maximum likelihood method since this method assumes the multivariate normality exists in the data (Gao et al., 2008). When this assumption is not met, various approaches can be adopted. These are:

a) Data analysis can continue using the maximum likelihood method since researchers have found that the method is robust enough to deal with some deviation from normality when sample size is large (Lei & Lomax, 2005). As non-normality increases, ML cannot determine correct estimators and a sample size greater than 500 is required to compensate for the non-normality and correctly calculate parameter estimates (Curran et al., 1996; Hoogland & Boomsma, 1998). However, maximum likelihood may not be appropriate with a critical ratio well over five and a sample size of 215.

b) Assess and remove outliers until normality improves enough to meet the assumption. However, it is crucial to consider the treatment of outliers since these are valid responses from participants and deleting them will result in the loss of important perceptions about the phenomenon under study. Also, deleting outliers results in fewer observations, resulting in reduced information and model power that can lead to the wrong model specification (Gao et al., 2008).

c) Use transformation methods to convert data into univariate normality. However, in most instances, these are ineffective and difficult to analyse. Osborne (2002) states that transformed variables increase complexity, making it difficult to interpret, and therefore, care must be taken when inferring the results of transformed data. Dago et al. (2021, p. 65) mention that "using transformations in general and logarithm transformation in particular can be quite problematic. If such an approach is used, the researcher must be mindful about its limitations, particularly when interpreting the relevance of the analysis of transformed data for the hypothesis of interest about the original data".

d) Use a method that is not sensitive to non-normal distribution. An example of this in AMOS is the asymptotically distribution free (ADF) estimation developed by Browne (1984). It has similar properties to ML but relaxes the normal distribution assumption. However, due to its CPU and memory intensive nature, it is more suited for relatively simple models (Bentler, 2006). Furthermore, it requires a considerable sample size of around 5000 to compute reliable estimates (Gold et al., 2003)

e) West et al. (1995) and Byrne (2010) mention bootstrapping is another approach to handling multivariate non-normal data. It is a very computer-intensive method whereby subsamples are randomly drawn with replacements from the original data, providing data for more accurate parameter estimates and fit indices. The Bollen-Stine bootstrap method can be used to assess model fit using bootstrapping in AMOS (Bollen & Stine, 1992). This method calculates an adjusted chi-square critical value after comparison with the original chi-square statistics generated by the ML method. When the Bollen-Stine bootstrap method produces an insignificant adjusted Chi-square ($p>0.05$), then there is a model fit.

Linearity

According to Field (2005), it is important that the relationships being modelled are linear, i.e., any changes in the independent variable will result in a constant change in the dependent variable. A curve estimation for all relationships in the model was done in SPSS and found to be sufficiently linear and significant to be tested using covariance-based SEM. A snapshot of these results illustrating the significant values for these results are documented in (Appendix D, Figure D.1).

Multicollinearity

Multicollinearity was assessed by analyzing the Variance Inflation Factor (VIF) and tolerance. If VIF is above 4 or tolerance below 0.25, multicollinearity might exist (Alhajeri & Safian, 2023). Appendix D, Table D.2 indicates that multicollinearity does not exist since VIF was below 4 and tolerance greater than 0.25.

Sample size

There is no definite consensus on the appropriate sample size for SEM. Some researchers suggest the number of respondents should be calculated per variable, with Bentler and Chou (1987) suggesting five respondents per variable and Nunnally (1967), cited by Ranatunga et al. (2020), indicating a minimum of 10 respondents per variable. Ding et al. (1995) suggest a minimum range between 100 and 150. Kline (2005) mentions a sample size of at least 200 respondents. The generally accepted norm for SEM is that the sample size should be approximately 200. The parameters for SEM suggested in Kline (2005) were used to guide the SEM analysis for the current study.

Confirmatory factor analysis

During the EFA exercise, 3 factors (people, facilitating conditions and habit factors) as well as items from other factors were removed (indicated in Table 7.5). This resulted in a revised model with factors listed in Table 7.8 and presented holistically in Figure 7.54.

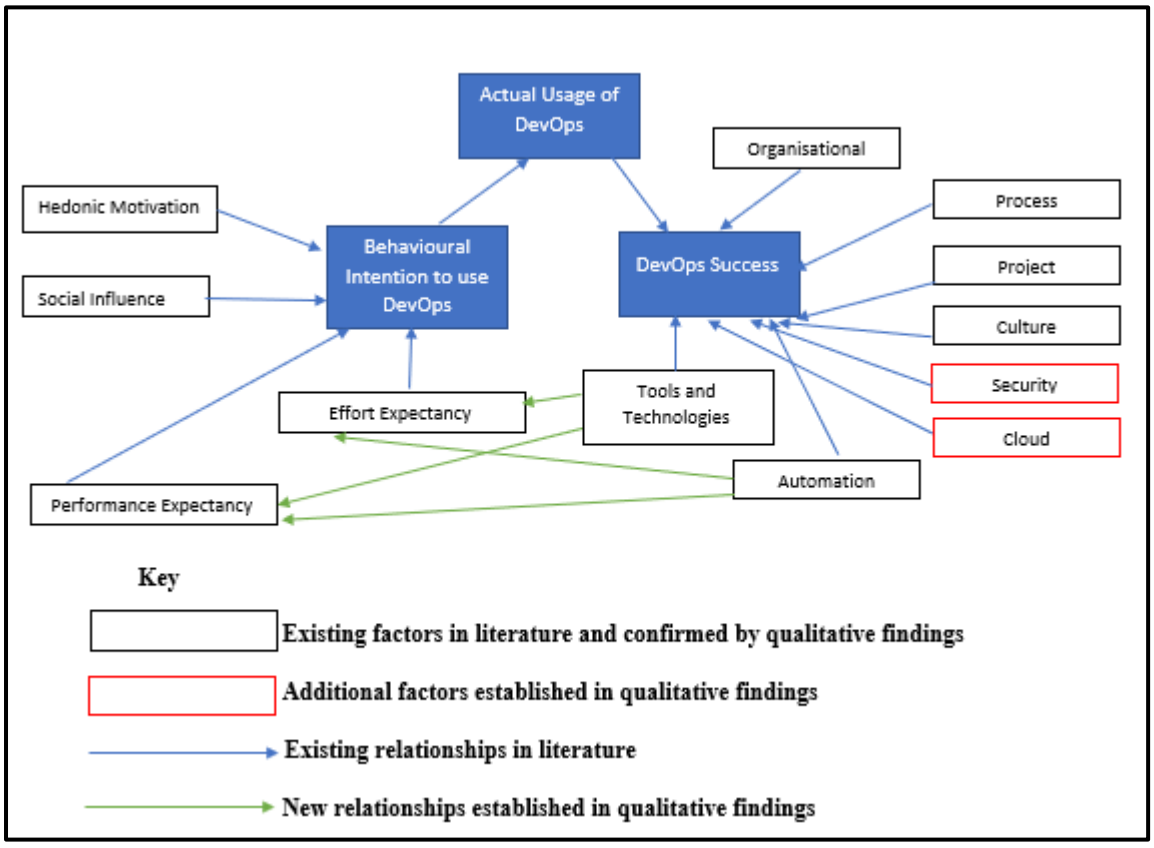

Figure 7.54: Model after EFA

Before testing this model's significant paths using path analysis in AMOS, it is important to ensure that the data fits the model. To achieve this, an essential step in SEM is to perform confirmatory factor analysis (CFA). Due to the data violating the multivariate normality, CFA was run using the maximum likelihood method with bootstrapping of 500 samples. Cheung and Lau (2008) stated that when the sample size is 200 and over, at least 500 to 1000 bootstrap samples should be generated. Furthermore, Deng et al. (2013) found little effect on the bootstrap standard error or confidence interval for bootstrap replicates ranging from 500 to 2000. In order to contextualise the presentation of the CFA results for the current study, a general discussion of the main test statistics used during CFA is presented.

Fit indices

Central to SEM is model fit, which demonstrates how well the measurement model and the structural model represent the data of the underlying conceptual model. Various fit indices assess model fit (Hooper et al., 2008). However, fit indices are not without their fair share of controversy and criticisms, with Barrett (2007) even elaborating on why they should be abolished. However, Marsh et al. (2004) allude to the usefulness of fit indexes but indicate that maintaining the strict cut-off values, as suggested by Hu and Bentler (1999) can result in a Type 1 error when a correctly specified

model can be rejected and in other instances, an incorrectly specified model is accepted. Many fit indices are grouped into absolute fit indices, Incremental fit indices and Parsimony fit indices. There are many indices, and reporting all is unrealistic and unnecessary. Hooper et al. (2008) suggested that the Chi-Square statistic, its degrees of freedom and p-value, the RMSEA and its associated confidence interval, the SRMR, the CFI and one parsimony fit index such as the PNFI should be reported since these indices are the least sensitive to sample size and parameter estimates.

Chi-Square statistic - Chi-Square (***χ2)*** statistics fall under the category of absolute fit indices. It is one of the popular ways of measuring overall model fit with a good model fit, resulting in an insignificant result when $p>0.05$ (Barrett, 2007). However, using this statistic alone to evaluate model fit is unreliable because it is affected by multivariate normality and large sample size (Bentler & Bonett, 1980; McIntosh, 2007). Research has also suggested that the relative/normed chi-square ($\chi 2/df$) can be used due to its low sensitivity to sample size variation (Wheaton et al., 1977). Research suggest a value as low as two (Tabachnick & Fidell, 2007) or as high as five (Wheaton et al., 1977) can indicate model fit. Hair et al. (2009) recommended a value of less than 3.

Root mean square error of approximation (RMSEA) - According to MacCallum et al. (1996), mediocre fits occur if the RMSEA test statistic indicates a value between 0.08 and 0.10 and a good fit is below 0.08. Stricter values of an upper limit of 0.06 (Hu & Bentler, 1999) or 0.07 (Steiger, 2007) have also been recommended.

Standardised root mean square residual (SRMR) - Diamantopoulos and Siguaw (2000) suggest that the values for SRMR can range from 0 to 1, with better-fitting models having a value less than 0.05. Hu and Bentler (1999) also state that values as high as 0.08 can be accepted.

Comparative fit index (CFI) - CFI is the most popular incremental fit index because it is not affected by sample size. A CFI of greater than 0.9 is suggested by Bentler and Bonett (1980). However, Hu and Bentler (1999) suggest a CFI value greater than 0.95. Researchers commonly refer to a CFI of 0.90 as a good fit and 0.95 as an excellent fit (Hair et al. (2009); (Montoya & Edwards, 2021).

Parsimony Normed Fit Index (PNFI) - Mulaik et al. (1989) established PNFI to compensate for higher fit values in models with many parameters. Hu and Bentler (1999) mentioned that a PNFI value of 0.5 or higher indicates an acceptable model fit.

Bollen-Stine bootstrap - *A*ccording to Bollen and Stine (1992), the Bollen–Stine bootstrap generates adjusted p values for model test statistics. Model fit is obtained when an insignificant p-value is obtained, $p>0.05$.

The Initial Measurement Model

The initial CFA measurement model for the current study is presented in Figure 7.55, with its corresponding fit indices shown in Table 7.8.

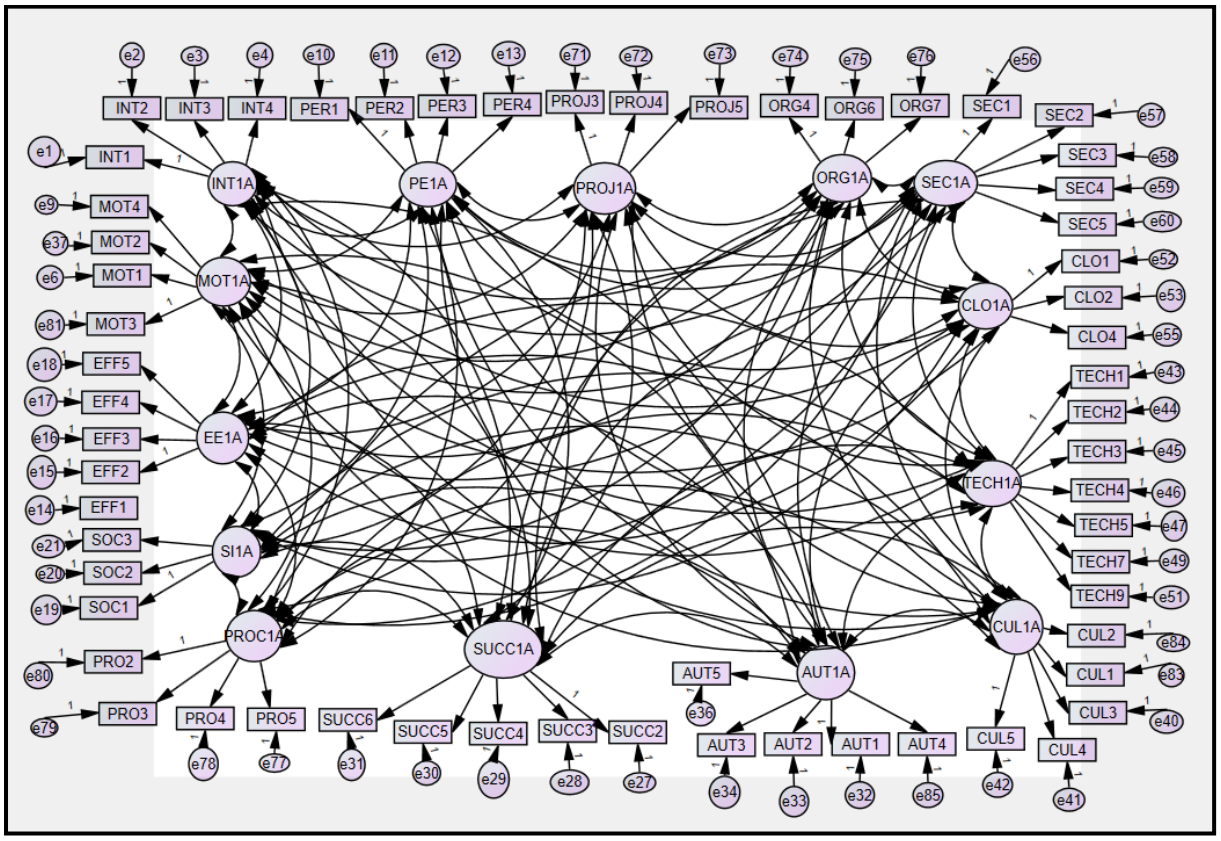

Figure 7.55: Initial measurement model

The labels of the latent variables shown in Figure 7.55 represent the following constructs: ORG1A – Organisational, SEC1A – Security, PRO1A – Process, PROJ1A – Project, TECH1A – Technologies, AUT1A – Automation, CLO1A – Cloud, CUL1A – Culture, SUCC1A – Success, PE1A - Performance Expectancy, EE1A - Effort Expectancy, MOT1A - Hedonic Motivation, SOC1A -Social Influence, INT1A – Intention.

The fit indices for the initial measurement model are shown in Table 7.8.

Table 7.8: Fit indices for the initial measurement model

Fit Index	Value	Acceptable/Not acceptable
Chi-square = 3362,127 Degrees of freedom = 1561 Probability level = ,000	Chi-square = 3525,456 Degrees of freedom = 1619 Probability level = ,000	Not acceptable, p <0.05
PCMIN/DF	2,178	Acceptable, < 3
RMSEA	0.074	Acceptable, < 0.08
CFI	0.818	Not acceptable, < 0.9
SRMR	0,0595	Acceptable, < 0.08
PNFI	0,651	Acceptable, > 0,50
Bollen-Stine bootstrap	0,002	Not acceptable, p < 0.05

The fit indices indicated that the model could be improved, especially in the context of CFI and the Bollen-Stine bootstrap values shown in Table 7.8. This was done by examining the standardised regression weights to determine if any weights were less than 0.5 (Hair et al., 2009). No regression weights less than 0.5 were identified. The standardised residuals covariance was also examined to determine if there was any values above absolute 2.58 (Byrne, 2010). No residual covariance was found above the absolute value of 2.58. The next step was to inspect the covariance between error terms (Byrne, 2010). Civelek (2018) further stated that when assessing covariance between error terms, it is essential not to place covariance between error terms that are not related in order to raise fit indices. Appendix D Table D.3 shows the error terms treated as free parameters, resulting in a decrease in chi-square (MAR change).

The final step to improve model fit was to analyse the modification indices regression weights (Kim, 2008; Wang & Ahmed, 2004). This resulted in the removal of the following items (ORG4, PROJ3, SEC4, SEC5, CLO4, TECH9, CUL1, CUL2, SUCC2, PRO4, EFF3, EFF5) since they were found to have high covariance with many other items thereby rendering these items as an unreliable measure (Snapshot of results Appendix D.4). This resulted in the final measurement model shown in Figure 7.56.

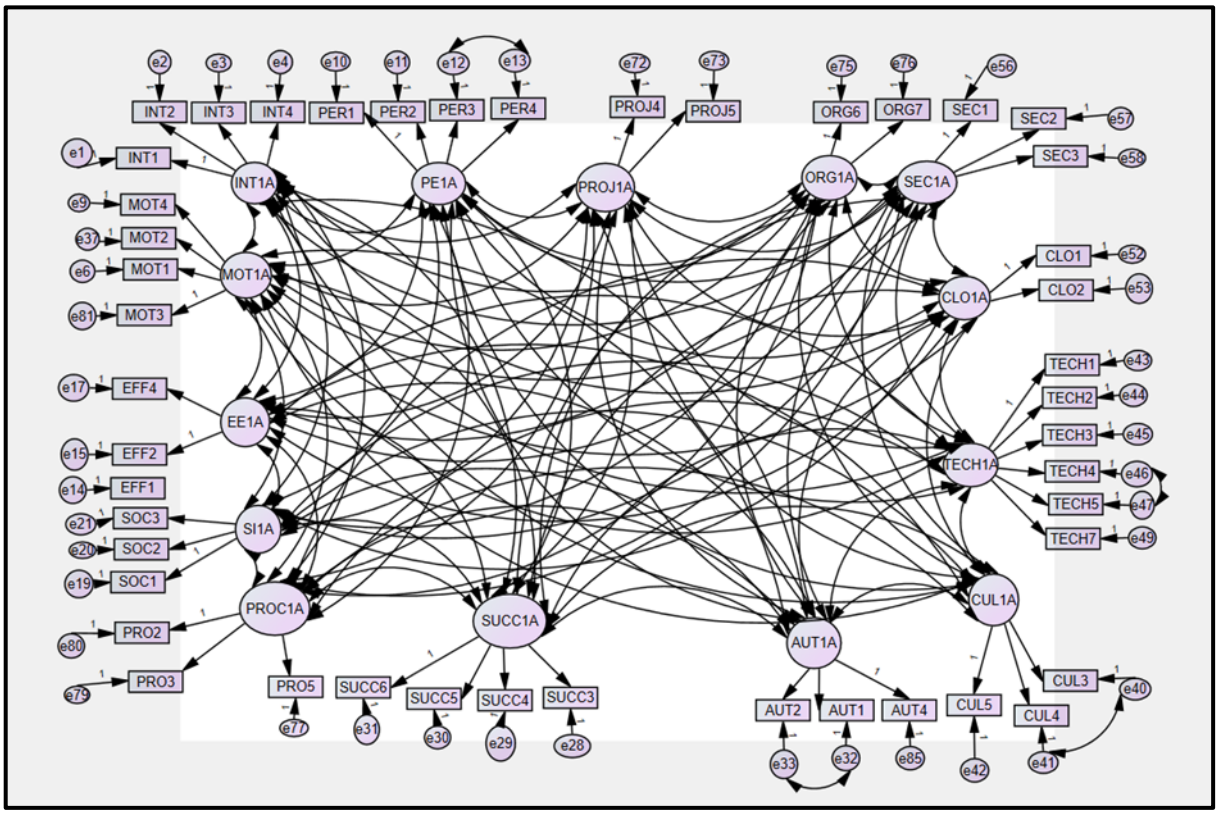

Figure 7.56: Final Measurement Model

The fit indices for the final measurement model are shown in Table 7.9. The table indicates that the modified measurement model (Figure 7.56) meets the minimum fit indices and can be evaluated for construct reliability and validity.

Table 7.9: Final measurement model fit indices

Fit Index	Value	Acceptable/Not acceptable
Chi-square Degrees of freedom Probability level	Chi-square = 1580,595 Degrees of freedom = 894 Probability level = ,000	Not acceptable, p <0.05
PCMIN/DF	1,768	Acceptable, < 3
RMSEA	0.060	Acceptable, < 0.08
CFI	0.906	Acceptable, > 0.9
SRMR	0.0487	Acceptable, < 0.08
PNFI	0.700	Acceptable, > 0,50
Bollen-Stine bootstrap	0.060	Acceptable, p > 0.05

Construct Reliability

According to Straub et al. (2004), construct reliability determines whether a set of variables (referred to as items in this research) consistently measures the construct under investigation. Cronbach (1951) mentioned that measurement-based research must be accurate, and reliability is a measure of this accuracy, with Straub (1989) defining it as "the extent to which the respondent can answer the same questions or close approximations the same way each time". Cronbach's alpha and composite

reliability are used to assess construct validity. SPSS V28 was used to calculate Cronbach's alpha. The composite reliability (CR) was determined using the standardised regression weights from AMOS together with an Excel spreadsheet (Analysis_Inn, 2020) containing the formula to calculate composite reliability (CR). Hair et al. (2009) indicate that a value of at least 0.7 should be obtained for both Cronbach alpha and CR for the construct to be considered reliable. Except for the project construct, all other constructs had a Cronbach alpha of above 0.7 and CR of above 0.7. The Cronbach alpha and CR were close enough to 0.7 for the project construct to accept that this construct displayed construct reliability (Table 7.10).

Table 7.10: Construct reliability and Convergent validity

Construct	Cronbach Alpha	CR	AVE
Organisational	0.732	0.734	0,58
Security	0.741	0,778	0,54
Process	0.756	0,759	0,512
Project	0.673	0,676	0,511
Technologies	0.893	0,898	0,597
Automation	0.857	0,868	0,689
Cloud	0.923	0.927	0,865
Culture	0.881	0,837	0,635
Success	0.884	0,884	0,657
Performance Expectancy	0.935	0,929	0,766
Effort Expectancy	0.853	0,857	0,667
Hedonic Motivation	0.861	0,87	0,629
Social Influence	0.926	0,93	0,816
Intention	0.938	0,938	0,792

Construct Validity

The ability of the items chosen to accurately reflect the construct is known as construct validity (Kline, 2005). For instance, construct validity will establish whether the six indicators used to test the technological construct measure this latent construct. Construct validity is established through convergent and discriminant validity (Byrne, 2010). Convergent validity establishes if all items that intend to measure a construct successfully determine the construct. Average Variance Extracted (AVE) is used to assess convergent validity. Bagozzi and Yi (1988) indicate that an AVE greater

than 0.5 demonstrate convergent validity since the indicators explain more than half of the variance in the latent construct. According to Table 7.11 all constructs meet the criterion of convergent validity. Discriminant validity establishes if all constructs in a study are distinctly different. This means that different constructs should not overlap when measuring the same concept. Discriminant validity is determined by the Fornell & Larcker Criterion, which is calculated by comparing the square root of AVE for a construct with the correlations of other constructs. If the square root of AVE for a particular construct is greater than the correlations of other constructs, then discriminant validity is confirmed (Fornell & Larcker, 1981). Table 7.12 indicates that all constructs had discriminant validity. Since all constructs displayed convergent and discriminant validity, all constructs were valid.

Table 7.12: Discriminant validity

	PE	EE	SI	INT	MOT	SUC	AUT	TECH	CUL	PRO	SEC	PROJ	CLO	ORG
PE	0,875	0.578	0,409	0,675	0,591	0,546	0,538	0,549	0,418	0,491	0,448	0,425	0,265	0,438
EE	0,578	0,816	0,504	0,613	0,609	0,46	0,389	0,481	0,403	0,496	0,546	0,449	0,226	0,501
SI	0,409	0,504	0,903	0,485	0,451	0,519	0,343	0,321	0,399	0,383	0,254	0,324	0,155	0,325
INTENTION	0,675	0,613	0,485	0,899	0,59	0,508	0,385	0,528	0,451	0,558	0,343	0,303	0,222	0,341
MOT	0,591	0,609	0,451	0,59	0,793	0,416	0,364	0,438	0,438	0,375	0,459	0,247	0,24	0,426
SUCCESS	0,546	0,46	0,519	0,508	0,416	0,81	0,695	0,674	0,595	0,566	0,454	0,507	0,316	0,366
AUTOMATION	0,538	0,389	0,343	0,385	0,365	0,695	0,83	0,674	0,519	0,543	0,515	0,463	0,48	0,411
TECHNOLOGY	0,538	0,481	0,321	0,528	0,528	0,674	0,674	0,772	0,537	0,611	0,527	0,362	0,395	0,429
CULTURE	0,418	0,403	0,399	0,451	0,438	0,595	0,519	0,537	0,796	0,565	0,468	0,437	0,193	0,349
PRO	0,491	0,496	0,383	0,558	0,375	0,566	0,543	0,611	0,565	0,715	0,727	0,587	0,303	0,671
SEC	0,448	0,546	0,254	0,343	0,459	0,454	0,515	0,527	0,468	0,727	0,734	0,531	0,339	0,575
PROJ	0,425	0,449	0,324	0,303	0,247	0,507	0,463	0,362	0,437	0,587	0,531	0,714	0,267	0,508
CLO	0,265	0,226	0,155	0,222	0,24	0,316	0,48	0,395	0,193	0,303	0,339	0,267	0,93	0,303
ORG	0,438	0,501	0,325	0,341	0,426	0,366	0,411	0,429	0,349	0,671	0,575	0,508	0,303	0,761

7.3.3 Structural Path Model

The final refined model during the EFA process was converted into a structural model (Figure 7.57). The structural model was obtained after the initial conceptual model was refined during EFA and CFA. The structural model's main dependent constructs (endogenous variables) are BI, usage and success. Since the usage construct consisted of a single item, it was not included during CFA. The independent constructs (exogenous variables) influencing success are automation, technology, cloud, security, organisation, people, process, and culture. The independent variables influencing BI are hedonic motivation, performance expectancy, effort expectancy and social influence. The structural model was analysed in AMOS using maximum likelihood and bootstrapping.

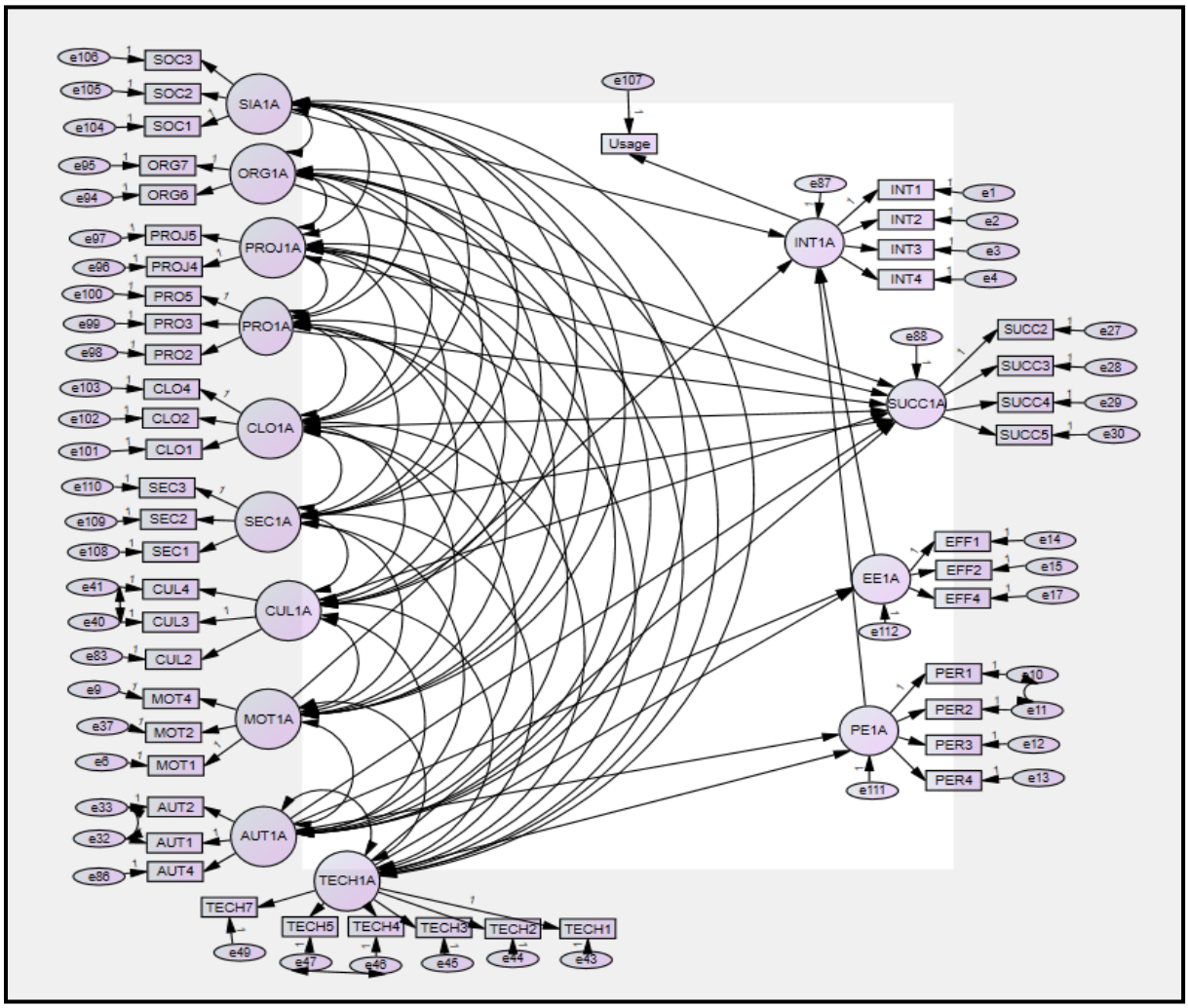

Figure 7.57: Initial Structural Path Model

The main test statistics emanating from the EFA model were observed to be within an acceptable range as shown in Table 7.13.

Table 7.11: Initial structural path model fit indices

Fit Index	Value	Acceptable/Not Acceptable
Chi-square Degrees of freedom Probability level	Chi-square = 1873,188 Degrees of freedom = 969 Probability level = ,000	Chi-Square ($\chi 2$) > 0.05 - Barrett (2007)
PCMIN/DF	1.933	PCMIN/DF <3 -Hair et al. (2009)
RMSEA	0.066	RMSEA < 0.08 - MacCallum et al. (1996)
CFI	0.879	CFI > 0.9 - Hair et al. (2009)
SRMR	0.064	SRMR < 0.08 - Hu and Bentler (1999)
PNFI	0,700	PNFI > 0.50 - Hu and Bentler (1999)
Bollen-Stine bootstrap	0,016	Bollen-Stine bootstrap > 0.05 - Bollen and Stine (1992)

While the model fit indices were within the expected ranges, it was observed that the model did produce paths that were recorded as not significant (p>0.05). These insignificant paths are shown in Table 7.13.

Table 7.12: Insignificant relationship in the initial structural path model

			Estimate	S.E.	C.R.	P
EE1A	←	AUT1A	,163	,117	1,394	,163
SUCC1A	←	ORG1A	-,131	,087	-1,510	,131
SUCC1A	←	PRO1A	,242	,170	1,421	,155
SUCC1A	←	CLO1A	-,025	,040	-,614	,539
SUCC1A	←	SECA	-,232	,102	-2,274	,123
Usage	←	SUCC1A	,109	,115	,946	,344

These insignificant paths may compromise the validity of the study's conceptual model. The following steps were taken to rectify this situation:

1) The insignificant paths shown in Table 7.13 were removed from the model.
2) The regression weights of the modification also yielded 4 new meaningful paths in the model, shown in Table 7.14.

Table 7.13: New links suggested by AMOS

			M.I.	Par Change
EE1A	←	MOT1A	24,986	,523
PE1A	←	MOT1A	20,922	,314
Usage	←	TECH1A	6,337	,306
Usage	←	AUT1A	5,198	,233

3) These new significant paths were added to the model, of which 3 were found to be significant (EE1A←MOT1A, PE1A←MOT1A and Usage←TECH1A). However, the path Usage←AUT1A was insignificant. Furthermore, the path INT1A←MOT1A, which was previously significant, became insignificant when the paths for EE1A←MOT1A and PE1A←MOT1A were added to the model.

Figure 7.58 shows the final structural path model, while Table 7.19 indicates its fit indices.

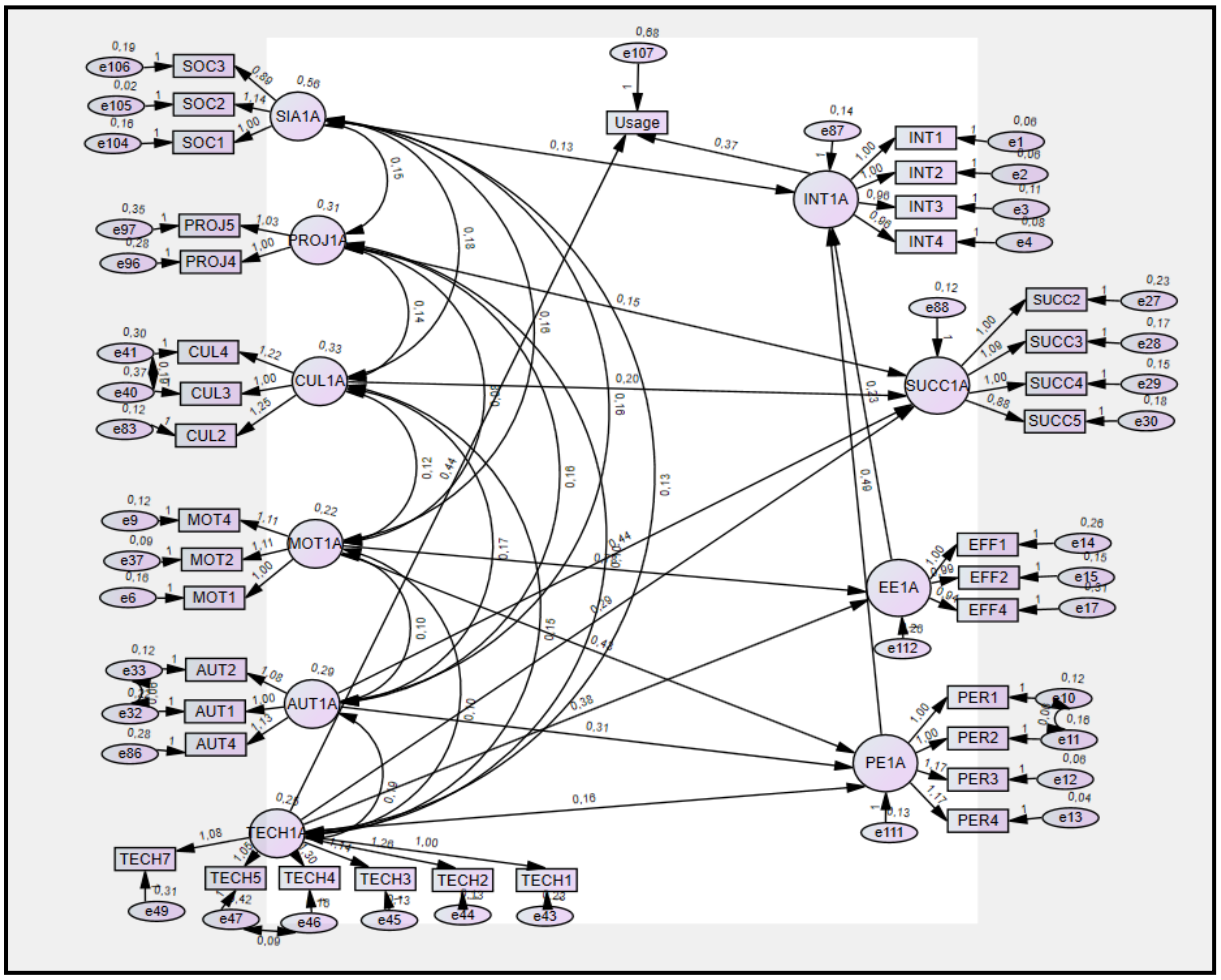

Figure 7.58: Final Structural Path Model

Table 7.15 shows that the fit indices are within the acceptable range, and the Bollen-Stine bootstrap value was insignificant, making the model acceptable.

Table 7.14: Final structural path model fit indices

Fit Index	Value	Acceptable/Not Acceptable
Chi-square Degrees of freedom Probability level	Chi-square = 984,483 Degrees of freedom = 562 Probability level = ,000	Chi-Square ($\chi 2$) > 0.05 - Barrett (2007)
PCMIN/DF	1,749	PCMIN/DF <3 -Hair et al. (2009)
RMSEA	0,059	RMSEA < 0.08 - MacCallum et al. (1996);
CFI	0,929	CFI > 0.9 - Hair et al. (2009)
SRMR	0.0592	SRMR < 0.08 - Hu and Bentler (1999);
PNFI	0,758	PNFI > 0.50 - Hu and Bentler (1999);
Bollen-Stine bootstrap	0,054	Bollen-Stine bootstrap > 0.05 - Bollen and Stine (1992)

7.3.4 Structural Model Assessment and Findings

A structured equation model generated through AMOS was used to test the relationships. A good fitting model is accepted if the value of the PCMIN/DF <3 -Hair et al. (2009); RMSEA < 0.08 - MacCallum et al. (1996); CFI > 0.9 - Hair et al. (2009), SRMR < 0.08 - Hu and Bentler (1999); PNFI > 0.50 - Hu and Bentler (1999) and the Bollen-Stine bootstrap > 0.05 - Bollen and Stine (1992). Each significant relationship established in the model is in Table 7.16.

Table 7.15: Standardised estimates of hypothesised relationship

Hypothesised Relationship	Standardised Estimates	t-value	P value	Decision
Culture → Success	0.208	2.763	0.006	Supported
Project → Success	0.189	2.378	0.017	Supported
Technology → Success	0.289	3.444	<0.001	Supported
Automation → Success	0.307	3.546	<0.001	Supported
PE → BI	0.462	6.933	<0.001	Supported
EE → BI	0.289	4.339	<0.001	Supported
SI → BI	0.174	3.060	0.02	Supported
BI → Usage	0.223	2.972	0.002	Supported
Technology → PE	0.208	2.407	0.016	Supported
Automation → PE	0.258	3.137	0.002	Supported
Technology → EE	0.271	3.711	<0.001	Supported
New Links				
Technology → Usage	0.244	3.147	0.02	
HM → PE	0.414	5.846	<0.001	
HM → EE	0.525	6.459	<0.001	
R-Square				
Success	0.637			
BI	0.524			
Usage	0.159			
PE	0.503			
EE	0.474			

The statistics presented in Table 7.16 are presented in a discussion format to enable easier comprehension,

Success

The squared multiple correlation was 0.637 for the success variable. This shows that 64% of the variance in success is accounted for by culture, automation, technology and project factors. The influence of automation on success was positive and significant (b = 0.307, t = 3.546, p < 0.001), supporting $H7_a$. This means that the value of the standard deviation of success will increase by 30.7% for every increase in one standard deviation of automation. The influence of Culture on Success was positive and significant (b = 0.208, t = 2.763, p = 0.006), supporting $H2_a$. This means that the value

of the standard deviation of success will increase by 20.8% for every increase in one standard deviation of culture. The influence of Technology on Success was positive and significant (b = 0.289, t = 3.444, p < 0.001), supporting $H6_{a.}$ This means that the value of the standard deviation of success will increase by 28.9% for every increase in one standard deviation of technology. The influence of Project factors on success was positve and significant (b = 0.189, t = 2.378 , p = 0.017), supporting $H5_a$. This means that the value of the standard deviation of success will increase by 18.9% for every increase in one standard deviation of project factors.

Behavioural Intention

The squared multiple correlation was 0.524 for the behavioural intention (BI) variable. This shows that 52% variance in success is accounted by performance expectancy (PE), effort expectancy (EE) and Social Influence (SI) factors. The influence of PE on BI was positve and significant (b = 0.462, t = 6.933, p < 0.001), supporting H11a. This means that the value of the standard deviation of BI will increase by 46.2% for every increase in one standard deviation of PE. The influence of EE on BI was positive and significant (b = 0.289, t = 4.339, p < 0.001), supporting H12a. This means that the value of the standard deviation of BI will increase by 28.9% for every increase in one standard deviation of PE. The influence of SI on BI was positive and significant (b = 0.174, t = 3.060, p =0.02), supporting H14a. This means that the value of the standard deviation of SI will increase by 17.4% for every increase in one standard deviation of SI.

Usage

The squared multiple correlation was 0.159 for the Usage variable. This shows that the BI and technology factors account 16% variance in Usage. The influence of BI on Usage was positve and significant (b = 0.223, t = 2.972, p = 0.002), supporting H10a. This means that the value of the standard deviation of usage will increase by 22.3% for every increase in one standard deviation of BI. During model assessment, a new link was indicated between usage and Technology. The influence of Technology on Usage was positive and significant (b = 0.244, t = 3.147, p = 0.02) resulting in a new link referred to as NL1. This means that the value of the standard deviation of usage will increase by 24.4% for every increase in one standard deviation of technology.

Performance Expectancy

The squared multiple correlation was 0.503 for the PE variable. This shows that 50% variance in PE is accounted by technology, automation and hedonic Motivation factors. The influence of technology on PE was positve and significant ($b = 0.208$, $t = 2.407$, $p < 0.016$), supporting H17a. This means that the value of the standard deviation of PE will increase by 20.8% for every increase in one standard deviation of technology. The influence of automation on PE was positve and significant ($b = 0.258$, $t = 3.137$, $p < 0.002$), supporting H18a. This means that the value of the standard deviation of PE will increase by 25.8% for every increase in one standard deviation of automation. During model assessment, a new link was indicated between PE and Hedonic Motivation (HM). The influence of HM on PE was positive and significant ($b = 0.414$, $t = 5.846$, $p < 0.001$) resulting in a new link referred to as NL2. This means that the value of the standard deviation of PE will increase by 41.4% for every increase in one standard deviation of HM.

Effort Expectancy

The squared multiple correlation was 0.474 for the EE variable. This shows that 47% variance in EE is accounted by technology and hedonic motivation factors. The influence of Technology on EE was positIve and significant ($b = 0.271$, $t = 3.711$, $p < 0.001$), supporting H19a. This means that the value of the standard deviation of EE will increase by 27.1% for every increase in one standard deviation of technology. During model assessment, a new link was indicated between EE and HM. The influence of HM on EE was positive and significant ($b = 0.525$, $t = 6.450$, $p < 0.001$) resulting in a new link referred to as NL3. This means that the value of the standard deviation of EE will increase by 52.5% for every increase in one standard deviation of HM.

7.3.5 Summary of the Study's Hypotheses

This study consisted of an initial set of 21 hypotheses listed in Table 7.17 together with an outcome that indicates whether each hypothesis was supported or not.

Table 7.16: A listing of the study's hypothesised relationships

Hypothesis	Final Findings
$H1_a$: *Organisational factors positively influence DevOps success in the South African software industry.*	Not Supported
$H2_a$: *Culture factors positively influence DevOps success in the South African software industry.*	Supported
H3a: People factors positively influence DevOps success in the South African software industry.	Not Supported
H4a: DevOps processes have a positive influence on DevOps success in the South African software industry.	Not Supported
$H5_a$: *Project factors positively influence DevOps success in the South African software industry.*	Supported
$H6_a$: *The use of technologies has a positive influence on DevOps success in the South African software industry*	Supported
$H7_a$: *Automation has a positive influence on the success of DevOps projects in the South African software industry.*	Supported
$H8_a$: *Cloud computing has a positive influence on the success of DevOps projects in the South African software industry.*	Not Supported
$H9_a$: *Security has a positive influence on the success of DevOps projects in the South African software industry.*	Not Supported
$H10_a$: *BI will positively influence the actual usage of DevOps.*	Supported
$H11_a$: *PE positively influences the BI of software professionals to use DevOps.*	Supported
$H12_a$: *EE positively influences the BI of software professionals to use DevOps.*	Supported
$H13_a$: *Facilitating conditions positively influences the BI of software professionals to use DevOps.*	Not Supported
$H14_a$: *SI positively influences the BI of software professionals to use DevOps.*	Supported
$H15_a$: *Habit positively influences the BI of software professionals to use DevOps.*	Not Supported
$H16_a$: *Hedonic motivation positively influences the BI of software professionals to use DevOps.*	Not Supported

$H17_a$: *Technologies positively influence PE in DevOps*	Supported
$H18_a$: *Automation positively influences PE in DevOps*	Supported
$H19_a$: *Technologies positively influence EE in DevOps*	Supported
$H20_a$: *Automation positively influences EE in DevOps.*	Not Supported
$H21_a$: *Actual usage of DevOps positively influences DevOps' success.*	Not Supported

Eleven of the 21 hypotheses were supported in this study. Technology, automation, culture and project factors were found to significantly influence *DevOps success*. *DevOps acceptance* was significantly influenced by social influence, performance expectancy and effort expectancy. Furthermore, 3 of the 4 additional paths established during the qualitative phase was also found to be significant. These were: technology positively influences performance expectancy, automation positively influences performance expectancy and technology positively influences effort expectancy. *BI* was also found to have a significant positive influence on DevOps usage. Also, during the structural path analysis, 3 more significant path were established. These are: Hedonic Motivation positively influences effort expectancy, Hedonic Motivation positively influences performance expectancy and Technology positively influences usage.

7.4 SUMMARY

The current chapter presented descriptive statistics on factors that influence DevOps acceptance and success. The descriptive statistics indicate that organisation, technology, automation, people, project, security, cloud and culture were important factors in *DevOps success*. Furthermore, performance expectancy, effort expectancy, social influence, habit, hedonic motivation and facilitating conditions have a positive influence on *DevOps acceptance*. Structural equation modelling was used to determine which factors where significant in determining the success and acceptance of DevOps. The significant factors for DevOps success were automation, technology, culture and project factors. The significant factors for DevOps acceptance were social influence, performance expectancy and effort expectancy. The next chapter synthesizes the qualitative and quantitative analysis.

CHAPTER 8: DISCUSSION OF FINDINGS

8.1 INTRODUCTION

A comprehensive analysis of the participants' perspectives on the concept of DevOps, as well as its advantages and disadvantages, was conducted in Chapter 5 of this research study. As an additional point of interest, the participants' personal experiences with DevOps offered valuable insights into the aspects that contribute to the success and acceptance of DevOps adoption. A qualitative research approach that included phenomenology, semi-structured interviews, and thematic analysis was utilised in order to accomplish this goal. The qualitative findings were incorporated into the preliminary conceptual model developed in Chapter 6, which led to the development of a revised model that included a set of hypotheses. Chapter 7 presented the descriptive and inferential statistics applied to the quantitative data to validate this new model and test the hypotheses. This chapter discusses and draws together the outcomes of Chapter 5 and Chapter 7. The research questions that are repeated here for ease of reference serve as the focal point for discussion.

1. What are South African software practitioners' perspectives on the DevOps phenomenon?
2. What acceptance factors influence software professionals' intention to use DevOps?
3. What success factors influence DevOps project success?
4. Which acceptance and success factors are critical to the adoption of DevOps?

8.2 PARTICIPANTS' DEVOPS DEFINITION, DEVOPS STRENGTHS AND CHALLENGES

The participants offered a variety of definitions of the term "DevOps." These definitions have been classified into the following themes: *culture of collaboration, the utilisation of tools, automation, processes and practices, people and ownership of code*. The culture of collaboration is essential since various stakeholders integrate their skills and knowledge to deploy software quickly into the operational environment. This collaboration was also alluded to by Farroha and Farroha (2014) when they defined DevOps as the overlapping of operational and developmental teams to improve the development process by creating a culture of collaboration. Participants also mentioned tools, automation, processes and practices, and people in their definition of DevOps. This aligns with the definition by Jonker (2017) that people, processes and technology enable both the rapid development and deployment of software. Participants furthermore mentioned that code ownership in their definition is essential in promoting the culture of shared responsibility. This is in keeping with Lwakatare et al. (2019, p. 224) definition that DevOps "means ownership and responsibility of the software development team to design, implement, test, deploy and maintain in production the

web applications and services". Participants in this study provided a diverse set of definitions for DevOps (similar to Maroukian and Gulliver (2020) and Lwakatare (2017)). While this variance in the definition of DevOps may be deemed to be somewhat lacking in any kind of consensus interpretation Lwakatare (2017) is of the opinion that this might be expected due to the multifaceted nature of DevOps. Based on the data collected, analysed and thematically presented in Chapter 5, a definition for DevOps is proposed as follows: *DevOps entails a merger between development and operations that results in a culture of collaboration that can be improved by utilising tools and automation within well-defined processes and practices.* This definition also has a strong alignment with the practitioners' assertion that one of the main purposes of DevOps is that it helps to maintain the quality of the software development process and establish code ownership.

In terms of the strengths of DevOps, participants identified both faster delivery and improved software quality as important elements. Participant 3 highlighted from his working experience that the development and delivery of a typical software system averaged approximately 112. With the advent of DevOps, the development and delivery cycle has been reduced to an average of 30 days. Rowse and Cohen (2021) concurred with these results by quantitatively measuring the perceptions of South African software professionals and found that 85% of respondents indicated that DevOps enabled more frequent builds in the DevOps environment. Participant 5 mentioned improved product quality due to better identification and implementation of non-functional requirements affirming Katal et al. (2019) findings. Another strength identified is the enhanced knowledge sharing in DevOps teams, echoing Nielsen et al. (2017) similar sentiment. Before DevOps, there was little ownership and accountability for the product developed, with the product being sent between the development and operations teams. This study found that a critical DevOps strength is the DevOps teams' shared ownership and accountability for the developed product. A similar perception that team ownership and accountability were critical in achieving the desired DevOps benefits was mentioned by Senapathi et al. (2018). Participants in this study further mentioned the earlier problem identification and resolution and greater visibility of projects as DevOps strengths.

The challenges identified by software professionals in achieving a successful DevOps environment were legacy code, too much focus on delivery, automated testing, not a one-size-fits-all process, the ability to unlearn and relearn, and cost. Due to the monolithic nature of legacy code, it is difficult to break it up into features and services, thus making it challenging to implement a DevOps approach in these environments. Alves and Rocha (2021) identified a similar challenge in their study Khan et al. (2022) and Ahonen (2020) also identified cost and automated testing as challenges to DevOps adoption. Unlearning and relearning were identified as challenging in adopting DevOps since professionals are unwilling to forget previous practices and technologies to learn new skills and processes. Khan et al. (2022) encountered a similar learning challenge in their study. This study also

found that DevOps is not a “one size fit all process” and it is vital to adapt the DevOps process to the type of project, organisational culture, and people skills. Colantoni et al. (2020) state that there are variants of DevOps frameworks, such as DevSecOps and AIOps, together with various technologies and therefore your choice needs to align with the project type and organisation. Too much focus on delivery is another challenge identified by participants. As much as DevOps teams want to achieve fast delivery, professionals indicated that this can become a challenge when considerable focus is on delivery only. This results in the team not reflecting on the current processes thus preventing them from optimising and maturing the DevOps processes.

8.3 PARTICIPANTS' PERCEPTIONS OF ACCEPTANCE AND SUCCESS FACTORS

8.3.1 Acceptance Factors

Performance Expectancy (PE)

According to the findings of the descriptive study, the respondents' perception of the PE of DevOps is high. The overall construct mean is 4.48 and the standard deviation is 0.58 indicating that the participants largely agree with the items that measure the PE construct. This is because the mean values are higher than the assumed neutral value of 3 on a scale that ranges from 1 to 5. The Wilcoxon Signed Rank test was used to verify the statistical significance of this result, thereby confirming a substantial consensus among software professionals that DevOps was beneficial.

The items *"Using DevOps increases my productivity within a DevOps team"* and "*I find DevOps useful for software development and deployment"* had the most positive responses.

The quantitative statistics for this construct corroborated the qualitative findings that allude to the usefulness of DevOps. The following verbatim quotes attest to this claim:

> *It is definitely useful. Before I use to develop software and give it to the operations division only to have come backs like incorrect dependencies files or missing files, etc. With DevOps and close alignment with operations, we develop more on production-like servers which means that the software has a less chance of failing in the operational environment* ***(Participant 6)***

> *Me as an individual has not seen the increased usefulness as yet in DevOps, since in testing we are still facing challenges with automating our testing process. But overall as a team, especially guys that are able to automate their processes, I would definitely said yes that DevOps increased the productivity of the team* ***(Participant 1)***

These results support the findings of a DevOps study by Anderson (2019), which indicated that respondents believed that DevOps was beneficial.

Effort Expectancy (EE)

The EE construct's mean value and standard deviation were 3.94 and 0.72, respectively. This suggests that respondents agreed that using DevOps was easy since the mean value was greater than the neutral value of 3. In addition, the findings of the Wilcoxon Signed Rank test demonstrated a significant consensus among software experts regarding the ease with which they could utilise and learn DevOps. The items with the most positive responses were recorded as: *"It is easy for me to become skillful using DevOps"*, *"I find DevOps easy to use when developing and deploying software"* and *"My interaction with DevOps is clear and understandable"*. This result also aligns with the qualitative findings where participants mentioned that DevOps was easy to use, especially if you are technically inclined and worked in a similar environment. The following interview extract reflects this:

> *I am a technical person. I love looking for new technologies to make my work more efficient and effective. For me, adjusting to the DevOps way of working was easy for me* ***(Participant 3)***

> *DevOps is one step forward from Agile. Agile enables more continuous development of software and joined with the DevOps approach, these features are implemented more rapidly into the operational environment. So DevOps is more aligned to the Agile approach that we currently use in this organisation since the practices and objectives are similar in both strategies. By already working in an agile way, it is easier to implement and adopt the DevOps strategy as opposed to working in the traditional waterfall environment* ***(Participant 4)***

These results support the findings of a DevOps study by Anderson (2019), which indicated that respondents perceived DevOps as easy to use.

Facilitating Conditions

The respondents were generally confident of the presence of facilitating conditions for DevOps adoption. A high mean of 4.06 and standard deviation of 0.69 suggest that most respondents agree with the items that measure the FC construct since it was greater than the neutral value of 3. The Wilcoxon Signed Rank test also indicates a significant agreement among software professionals that they have access to the required resources and knowledge to use DevOps. The items with the highest agreement were recorded as: "*DevOps was compatible with their other software development processes that I use"*, *"I have the knowledge necessary to use DevOps strategy for software development"*, and *"I have the resources necessary to use DevOps"*. The qualitative finding also supports this quantitative conclusion where participants indicated that they have the necessary FC to use DevOps as demonstrated by the participants' interview quotations below:

> *My organisation has invested a lot of money into the DevOps approach to ensure that it is successful. Management matched the vision of DevOps with increasing*

the resources necessary for it to succeed. To gain the maximum benefit of DevOps, you need to promote the use of tools and automation, which our organisation invested in and promote. Having these conditions are important for the DevOps strategy to be accepted and be successful ***(Participant 2)***

For DevOps to be successful, it takes more than collaboration. Although collaboration is the start of the journey for DevOps to be successful, technologies are a must, and our organisation invested in these technologies so DevOps teams have the necessary infrastructure for project success. Besides spending on technologies and infrastructure, it also makes training available so DevOps teams are knowledgeable on the processes and tools ***(Participant 1)***

These results support the findings of a DevOps study by Anderson (2019), which indicated that respondents agreed that they have the necessary resources and support to use DevOps.

Social Influence

Descriptive results indicated that respondents felt that others influenced them to use DevOps. A mean value of 3.96 and standard deviation of 0.78 for this construct reflects that most respondents agreed that others influenced them to use DevOps. The Wilcoxon Test provides further statistical evidence, indicating significant agreement among software professionals that others influenced their use of DevOps. Items with the highest agreement were recorded as: *"People whose opinions that I value prefer that I use DevOps for software development"*, *"People who influence my behavior think that I should use DevOps for software development"*, *"People who are important to me think that I should use DevOps for software development"*. Triangulation of the quantitative results with the qualitative findings further supported the finding that participants indicated that others influenced DevOps use.

There is always a degree of influence from others. You have friends in other organisations and they speak about how efficient their software processes are due to DevOps and you do get influenced by their reviews. ***(Participant 4)***

Let face it, if you are IT professionals, especially in software development, tools are changing rapidly and you must be able to learn and innovate to keep up with these changes. Interaction with people outside your organisation certainly helps to keep you on your toes. Also, I work in the banking sector. It is highly competitive with banks competing against each other to be the most innovative and come up with new products to attract and maintain customers, especially on the online platform. So, you do get influenced by what your competitors are doing ***(Participant 8)***

These results support the findings of a DevOps study by Anderson (2019), which indicated that respondents agreed that others influenced them to use DevOps.

Habit

The mean for this construct is 4.3 and the standard deviation is 0.61 suggesting that the participants agreed with the items measuring this construct. This indicated that respondents strongly agreed that using DevOps became a habit for problem-solving, automating processes, and collaborating. This agreement can also be noted from the significant results of the Wilcoxon Signed Rank test. The items with the highest agreements were recorded as: *"It became a habit to help team members in other domains to fix things with a project when it fails"*, *"It became a habit to look for ways to automate my software development processes"* and *"It became a habit for me to collaborate with teams members from other IT domains"*. Qualitative responses also supported the quantitative finding, with participants mentioning that working in a DevOps environment became a habit as demonstrated from the extracts below:

> *Initially DevOps was just an inconvenience in my life. Going to meetings, collaboration with other teams, etc for me was a waste of time. I just wanted to get my work done. However, when I saw the benefits that was achieved in terms of quicker development, more quality software and less problems, DevOps became part of my daily routine* ***(Participant 5)***

> *Worked on few features using DevOps and it became a habit to collaborate with others and solve problems as a team with diverse skills and knowledge. It also became a habit for me to continuously see how I can improve the testing in DevOps for my team* ***(Participant 1)***

Hedonic Motivation (HM)

A high mean of 4.33 and a standard deviation of 0.53 indicate that most respondents agreed with the items that make up the HM construct. These statistics suggest that the respondents agreed that that the use of DevOps was enjoyable. The statistical significance of the Wilcoxon Signed Rank test further reflects this agreement. Items with the highest agreements were recorded as: *"have fun playing with and investigating new tools to improve the DevOps strategy"* and *"motivated to work since things get done faster and problems are solved more efficiently and effectively"*. The quantitative findings mirrored the qualitative findings, where participants indicated pleasure and job satisfaction using DevOps. This is evident from the participants' quotes below:

> *The operation guy should always take the hit with a feature does not work in the operational environment. Something goes wrong and you contact the developer and his response is that I gave you a working feature and now I am too busy on another project to look at it. Previously it was the classic case of throwing the code to the operations department and not taking any responsibility thereafter. However now with greater teamwork, there is no blame game and this interaction is better and work more enjoyable* ***(Participant 7)***

> *DevOps definitely improved my job satisfaction. Being in security, you want to be consulted early on what security risks exist and how they can be prevented when developing the app. With DevOps you are part of the multidisciplinary team and we are able to identify and solve problems sooner. You enjoy your work more* ***(Participant 2)***

Behavioural Intention (BI)

Descriptive statistics for the BI construct indicate that the majority of the respondents agreed with all of the constructs. A mean of 4.3 and a standard deviation of 0.53 suggest that respondents clearly have an intention to use DevOps for software projects. The items with the highest agreement were recorded as: "*I predict that I will be able to use a DevOps approach for software development projects in my organisation"*, *"I intend to make an effort to practice a DevOps approach for software development projects on a regular basis"* and *"I intend to use a DevOps approach for software development projects on a regular basis"*. The high BI mean and high mean for all the acceptance factors indicate that acceptance factors may significantly influence the BI to use DevOps.
These results support the findings of a DevOps study by Anderson (2019), which indicated that respondents agreed that they intend to use DevOps.

Actual Usage

A majority (76.7%) of the participants indicated that they either used DevOps "almost always" or "often". The BI construct and the FC construct both had a high mean. This may indicate that FC may significantly influence actual usage.

8.3.2 Success Factors

Organisational

The mean and standard deviation computed for this construct are 3.67 and .0.61 respectively, indicating that a majority of the respondents agreed that the organisational factors influence the use of DevOps. The preceding claim is supported empirically by the statistically significant Wilcoxon Signed Rank test. The items that received the highest agreement were recorded as: *"DevOps project team received good management or executive support"* and *"organisation has a cooperative culture instead of hierarchical"*. In this regard, the quantitative perception of respondents matched the views given by participants in the qualitative findings, with participants justifying the importance of organisational factors for DevOps' success. The following extracts supports this:

> *Visibility is important there. If it is from the upper management support I think visibility would play a key... for my my view key aspect to say that we are part of the journey. Not just send out a comms. We are part of the journey. We see what you guys are going through and we know what needs to be done and we*

support that. Because to be honest teams are always led by a captain and it would be nice now and then that you can see your captain is also part and parcel of this journey and your captain understands where you guys are at ***(Participant 1)***

I also think that previously, the developer was at the bottom of a hierarchal type structure and the hierarchical people above didn't have a clue what they did, we were just managers or whatever, and the DevOps type methodology and Agile has brought the developer back more to the centre again. Developers are more involved with the decisioning and what have you. Previously, it was the manager that made the decision whether he went live or not, and sometimes a manager would say, you will go live and a developer would say, 'no.' Then they go live and there's a disaster so, yes, I think that's a very big change that has actually changed the organisation of… Giving more power or autonomy to the developer and having the developer's input previously, you will deliver on this day, we are going live on this day, and that's it. It didn't matter what but the developer just had to attempt to deliver. It's much more, better ***(Participant 4)***

However, only 39.6% agreed that DevOps teams should be collocated working from the exact location. This contradicted some of the participants' qualitative responses, who felt strongly that co-location and face-to-face physical interaction were necessary for DevOps success.

Look, co-location is absolutely critical more than ever. It's about collaboration, cross-pollination, you talk about the X-type resources... it's very important. ***(Participant 3)***

From an infrastructure point of view, we've worked like DevOps and Agile approach if we needed to sort something out... we've always been two space housing and security and also, the SDLC environment. You grab the right people together in a room quickly, sort out what you need to do, and off you go. You don't try and send off mails... you go to the person. So, that's just how we've always worked. ***(Participant 5)***

A possible reason for this was that the interview data was collected during the prior to the period of the COVID-19 pandemic, where software professionals were collocated daily. However, most of the quantitative data was collected during the period of the COVID-19 pandemic, when most professionals worked remotely. This resulted in them providing negative responses to the co-location item after experiencing their ability to work from home and communicate using platforms such as Zoom or MS Teams.

These results support the findings of a DevOps study by Azad (2022), which indicated that organisational factors play an important role in DevOps success.

People

Regarding the people factor, a majority of the respondents agreed that people-oriented factors were crucial for DevOps success. The preceding assertion is supported empirically by a reported mean value of 3.86 and a standard deviation of value of 0.56. A statistically significant Wilcoxon Signed Rank test further substantiated this claim. Items for this construct with the highest agreement were recorded as: "*DevOps team members are willing to learn and innovate"*, *"DevOps team members are highly motivated"* and *"team members have a high level of technical competence and expertise"*. The triangulation of qualitative and quantitative statistics for this construct provides further evidence that people factors are important for DevOps' success. The following verbatim quotes are indicative of this:

> *It is the future. It is the future but it should not be the silver bullet. It shouldn't be the silver bullet. At the end of the day it is just a process, it is a methodology. It is the people that make it work and if you don't have the right people with the appropriate skills or you not looking after your people, you not getting your people involved into what you are trying to sell them it is not going to work* ***(Participant 4)***

> *Well, you need to have passionate people around. So, we get a lot of documentation and design and all of that but you need to have people that wants to sit down and solve the problems to move forward. So, you have your journey but to work out the technical implementation and then find those guys and then when those guys start talking to each other then things start happening* ***(Participant 2)***

These results support the findings of a study on software development professionals who agreed that the people factors are important in software development project success (Bogopa & Marnewick, 2022).

Project

The majority of the respondents agreed with all of the items included in the project construct. The respondents agreed that project factors are important in DevOps projects, as indicated by a mean score of 3.68 and a standard deviation of 0.56. A significant Wilcoxon Signed Rank Test provided empirical evidence to confirm the validity of these aggregate values. Items that had the most agreement were recorded as: "*DevOps projects are characterised as mission-critical business software"*, *"DevOps projects have a variable scope that leads to new system requirements"* and *"DevOps projects have a dynamic, accelerated schedule"*. Further proof that project factors are essential for the success of DevOps can be found through the triangulation of both the qualitative and quantitative findings for this construct. The following verbatim quotations demonstrate this:

So, yah, business-critical software. DevOps is most appropriate for projects that are customer-facing. You want as little downtime as possible with customer-facing applications. DevOps is, therefore ideal for these applications because you want them deployed quickly and problems resolved with them as soon as possible. ***(Participant 8)***

Projects giving the organisation a competitive advantage are ideal for DevOps since organisations want the software delivered and deployed rapidly. Furthermore, using DevOps the project quality is better and less prone to errors in the operational environment. ***(Participant 4)***

These results support the findings of Ramadan and Rizk (2015) indicating that people factors are important for Agile software development project success.

Process

Almost all of the individuals who participated in the survey expressed their agreement with all of the items of the process factor. A mean score of 4.08 and a standard deviation of 0.58 were recorded for the Process construct, indicating that the respondents agreed that process factors play an essential role in the success of DevOps. A significant Wilcoxon Signed Rank Test provided empirical evidence to confirm the validity of these aggregate values. The items that received the most consensus were recorded as: *"DevOps projects continuously integrated team members' work, thereby reducing risk and identifying issues early in the software development life cycle"*, *"DevOps projects follow the practice of continuous delivery ensuring that the software can be released any time"*, *"DevOps projects were monitored continuously for early warning operation and quality defects that may occur in production"*. Comparing the results of qualitative and quantitative findings for this construct provided further confirmation that processes are vital for the success of DevOps. The following quotations from participants support this:

Some important principles and practices in the DevOps framework exist, such as continuous delivery, deployment, testing and monitoring. These principles and practices can only be implemented within the processes in the DevOps lifecycle. If these processes are not implemented then the core practices of DevOps will be missing and the benefits of DevOps will not be attained. Therefore it is important to follow the DevOps process to achieve project success. ***(Participant 8)***

The DevOps processes are important to the success of DevOps. Just like any methodology or framework, there are processes embedded so they can be successful when implemented. If we do not have these processes there can be chaos within the project because everyone will be doing their own thing. However, still got a lot to do in getting these processes fine-tuned using automation, etc. ***(Participant 5)***

These results support the findings of a study on software development professionals who agreed that the process factors are important in software development project success (Bogopa & Marnewick, 2022).

Culture

The descriptive statistics for this construct indicate that the majority of the respondents agreed with all of the items included. A mean score of 4.19 and a standard deviation of 0.69 were recorded for the Culture construct, indicating that the respondents agreed that culture plays an essential role in the success of DevOps. A significant Wilcoxon Signed Rank Test provided empirical evidence to confirm the validity of these aggregate values. The items with the highest agreement were recorded as: "*DevOps teams have a culture of collaboration"*, *"DevOps team have a culture of automating almost everything"* and *"DevOps team have a strong focus on continuous improvements to optimise performance, cost and speed of delivery"*. Aligning the quantitative results with the qualitative findings further corroborates that culture plays an essential role in DevOps.

> *... My problem is, and now I'm going to sound like a more senior veteran, but the problem is as the new kids came in, they found that level of automation without really understanding what happens behind it. It just 'poof' and magically it appears. But they don't really understand what's the science behind it. The people that knew the science moved on. So, now you're left with these kids that basically are just, with respect, because that's where I started my journey almost 20-years ago, as an operator. But they're basically operators. Now, all of a sudden, you say from tomorrow you will be an engineer and you now need to automate your infrastructure stack. Typically, a lot of our guys today, they can't even do normal bash scripting because they found it already there. Sorry, I made a long story about it, but skill and culture.* ***(Participant 3)***

> *. Everybody you know if you working in a DevOp organisation everybody should be held at the same level in terms of responsibility, in terms of seniority, in terms of delivery, we shouldn't be creating hierarchies and I think that is what the whole DevOp thing is trying to get rid of creating hierarchies because we need to live as a team. Not as a him and her she or they. It shouldn't be you know it's the development's problem. So if I have a defect for example it shouldn't be I'm sending the defect to the Dev and saying ah meeting you with the finger because you called it badly, you need to fix it and I'm not going to allow you to go into productions. Should be a thing where if we find a problem we deal with it together and we get the resolution together so I am able to assist much faster. Not just throw it over the fence or saying ah no you know what it doesn't work. We will put it in production and then we will sort it* ***(Participant 1)***

These results support the findings of Rowse and Cohen (2021) indicating that culture is important for DevOps success.

Tool and Technologies

A high mean of 4.26 and a standard deviation of 0.61 indicate that the majority of respondents agreed with all the items that make up the technology construct. A significant Wilcoxon Signed Rank Test provided empirical evidence to confirm the validity of these aggregate values. Items with the highest agreements were recorded as:*"DevOps projects use version control tools (such as Git and Github) to coordinate work among programmers and that DevOps projects use planning tools (such as Jira) for planning and project tracking allowing the team to ship code early and often"*, *"DevOps projects use continuous integration and delivery tools like Jenkins"* and *"DevOps projects use builds tools (such as Gradle or Maven) to automate builds, allowing for automatic download and configuration of dependencies or other libraries"*. Comparison of the qualitative findings with the quantitative also indicated that participants perceived that technologies are important in DevOps projects. Examples of qualitative findings demonstrating this are documented below:

> *You must have the right tools. Some... the right tools for the technology that you are working on. Technology should be your friend. Tools should be your friend as well. So often if we don't have the right tools to deliver something faster you would end up... you would end up just ruining the whole methodology process you want to do. So technology and the people as well as the risk play an important role* ***(Participant 1)***

> *So, in terms of tooling, definitely the DevOps is to automate your infrastructure provisioning. Without tools you can't do that so, yes, tooling is definitely an enabler for it. The choice of tooling is not as important as the process that you're solving. I don't have any holy cows, whether I use Bamboo or Jenkins... it's not as important as the script to write to provision the infrastructure. Now, whether that script is in Bash, or Chef... as long as it's understandable and sharable, that's what it's about.* ***(Participant 2)***

These results support the findings of a DevOps study by Azad (2022), which indicated that technology plays an important role in DevOps success..

Automation

An overwhelming majority of the respondents in the survey expressed their agreement with all of the items related to the automation factor. A high mean of 4.17 and a standard deviation of 0.64 indicate that the majority of respondents agreed that DevOps projects use automation. A significant Wilcoxon Signed Rank Test provided empirical evidence to confirm the validity of these aggregate values. The items that received the most consensus were recorded as: "*DevOps projects automatically provision for servers and installation of application code*", "*DevOps projects use an automated environment configuration for integration*", "*UI and regression testing and DevOps projects automatically*

provision for servers and installation of application code". Triangulation of this construct's qualitative and quantitative statistics further proves that automation is vital for DevOps success. Below are some quotes from participants indicating this:

> *So automation it's certainly improved the processes. It might make certain functions or roles redundant. But it would then open up opportunity for other focus points, without a doubt (**Participant 9**).*
> *I have to say that automation also plays a very significant role in getting us to get things in faster. But automation only works if your code is stable. Especially in my environment if you got a stable code base where Dev has done its work in terms of they've actually automated their integration testing as well. It just makes it so much easier when we press the button and the thing works. (Participant 1)*
> *I know for a fact the moment I started practising SaFe I need to start looking into DevOps. I need to start automating my tasks and at the end of the day I need to deliver quality and I need to deliver quality more frequently. What helps me deliver quality more frequent, might not necessarily be manual processes but a lot of automation where I can. So if you can automate a lot of the processes then you have to go for it. **(Participant 6)***

> *Automation, the ability to scale . To be able to sustain that sort of demand is self-service with automation. So traditionally in organisations like us if you wanted a server, you log a ticket. If you wanted somebody to look at your server and what's going on you log another, you log a ticket and you get into the queue, alright. Today, certain, certain platforms, we allow developers to build their own servers, that's why I kind of mentioned, you can build servers in two to three to five minutes. They, they are preallocated a certain resource pool and they can build against that and that's how we work. So I probably think that self-service that that's probably the most important part of it. **(Participant 8)***

These results support the findings of a DevOps study byMishra and Otaiwi (2020), which indicated that automation play an important role in DevOps success..

Cloud

The descriptive statistics for the Cloud construct indicate that the majority of the respondents agreed with all of the items included. The mean was recorded at 3.87 and the standard deviation was recorded as 0.81. A significant Wilcoxon Signed Rank Test provided empirical evidence to confirm the validity of these aggregate values. The items with the highest agreement were recorded as:*"DevOps projects use Infrastructure as Code for the provisioning of servers and installation of application code"*, *"DevOps projects use cloud platforms to provide advanced automation and DevOps projects use a centralised cloud platform for testing"* and *"deployment, monitoring and operating the production workloads"*. Aligning the quantitative results with the qualitative findings further corroborates that the cloud plays an essential role in DevOps. Evidence of this is provided below:

Cloud computing is huge and it's the buzzword and it makes sense in so many ways, it's not even funny. The we have or the level of automation and maturity that we've achieved in some of our teams already. Whether it runs public cloud or internal cloud... it's irrelevant. Everything is code and Agile aside, the DevOps culture and mindset in terms of the full-stack engineering methodology, automation... yes, that just makes the transition to cloud so much more seamless. ***(Participant 3)***

In terms of a DevOps team like the DevOps guys that are sitting on the fifth floor that's looking at AWS and what it means. For them, the success is to be able to have the teams have as much freedom to be able to create the environments that they need, without much intervention from them that they can basically manage the process, I mean the actual service provider, the infrastructure with it, that is basically to be able to consume those DevOps' requirements from the teams as easily as possible and without letting the team break everything else. ***(Participant 9)***

These results support the findings of a DevOps study by Mohammad (2018), which indicated that the cloud play an important role in DevOps success..

Security

The majority of the respondents agreed with all of the items included in the Security construct. The respondents agreed that security is considered in DevOps projects, as indicated by a mean score of 4.02 and a standard deviation of 0.64. A significant Wilcoxon Signed Rank Test provided empirical evidence to confirm the validity of these aggregate values. Items that had the most agreement were recorded as: *"DevOps projects use code analysis for vulnerability testing and quality assurance"*, *"DevOps projects use a change management process to ensure that any developer can suggest a mission-critical security change"*, and *"DevOps projects compared new features and updated against evolving threats"*. Further proof that project factors are essential for the success of DevOps may be found through the triangulation of both the qualitative and quantitative findings for this construct. The following verbatim quotations demonstrate this:

Since DevOps produce features at a rapid speed, developers can lose sight of the security aspects that need to be considered and therefore it is important that security be in all phases of the lifecycle, which means that a security personnel need to be part of the team. Sometimes, this may not be possible since the number of development teams in an organisation significantly outnumbers security specialists but it is important that feedback flows between them. ***(Participant 2)***

The only thing I can add is the process we started with. The last thing we'll take away from that is security was involved from inception. Whereas, we've been part of other projects but only much further down the line, and literally at the 11th hour, and you're expected to pull a rabbit out of the hat because they're

going live that weekend, but you were never part of the design or initial meetings, etc., whereas this, we were part and parcel from the beginning to end and it makes a big difference. ***(Participant 5)***

These results support the findings of a DevOps study by Akbar et al. (2020), which indicated that security play an important role in DevOps success..

Actual Success

The majority of the respondents agreed with all of the items included for the success construct. The preceding claim is corroborated by high mean value of 4.1 and a standard deviation of 0.6 that was recorded. A significant Wilcoxon Signed Rank Test provided empirical evidence to confirm the validity of these aggregate values. Hence, respondents perceived DevOps to be successful. The following items had the highest agreement: "*DevOps projects are successful in terms of deployment success rate*", "*DevOps projects are successful in terms of meeting the customer/end user expectations of the system and in terms of quality of the project outcome or of the resulting software project*" and *"DevOps projects are successful in terms of the speed of deployments"*. Furthermore, participants during the qualitative phase also mentioned DevOps success, as indicated by the quotations below:

My view is obviously... if you look at the traditional way of doing things. Which is waterfall it took too long and my view of DevOps is how we are able to resolve the mess that happens in IT organisations in delivering widgets if I may use the word, to the customer. It is important and the only way you can do it is if you do it much faster, better quality and something that is for the customer. You can't just give a customer a widget that the customer didn't want and the mere fact with the DevOps, the whole DevOps journey is you are more able to be more close to what the customer is wanting. And if I'm saying the customer, that is the business because the business is the one that gets the information from the customer. So if you are able to playback to the business which is the eyes and ears for the customers much quicker and deliver much faster it could show the customers that this is essentially what the DevOps is about ***(Participant 1)***.

I think we've got the stats to back it up. It's definitely made a huge difference. It's really dramatic what we've seen in terms of the average, and please, I don't have the exact days at hand but it was something like, on average, a couple of years ago it took 112-days, for example to land a typical project. Where I think last we checked it was 30-odd days now so, that in itself, speaks volumes. I believe it's been successful. I also like the saying where I don't believe we are where we want to be. Nowhere close, but we're also not where we used to be. Yes, we make mistakes. We've got a lot to learn still but the trajectory is heading in the right direction ***(Participant 3)***.

As just mentioned, respondents' perception of actual success was high. Comparing the means of the success factors showed means well above three, and the Wilcoxon Test for all these constructs was significant. Therefore, we may assume that these factors are significant in DevOps success.

8.4 CRITICAL SUCCESS AND ACCEPTANCE FACTORS FOR DEVOPS ADOPTION

The structural equation modelling (SEM) analysis discussed in Chapter 7 was used to determine the significance of the acceptance and success factors influencing BI and DevOps usage. These results will be discussed and compared with findings from related studies.

8.4.1 Influence of Success Factors on Actual Success

Results from this study indicate that automation, technology, culture, and project factors significantly influence DevOps' success.

Automation

A positive significant relationship between automation and success was established using SEM. Therefore, automation positively influenced DevOps success. The items after exploratory factor analysis (EFA) and confirmatory factor analysis (CFA) that were significant in achieving this result were:

- *DevOps projects use automated execution for continuous testing*
- *DevOps projects use an automated build trigger for continuous integration*
- *DevOps projects use automated performance and error notifications during continuous delivery*

These items indicate that automated execution of testing, automated build triggers, and automated performance and error notifications are essential for DevOps' success as found in studies by Gupta et al.(2017) and Perera et al. (2016).

Furthermore, the beta coefficient of 0.307 indicates that this construct is the most critical factor for DevOps success.

Tools and Technologies

SEM results indicated a positive significant relationship between technology and success. Hence, technology positively influenced DevOps success. The items after EFA and CFA that were significant in achieving this result were:

- *The DevOps projects use planning tools (such as Jira) for planning and project tracking allowing teams to ship code early and often.*
- *The DevOps projects use continuous integration and continuous delivery tools (such as Jenkins)*

- *The DevOps projects use Version Control tools (such as Git and GitHub) to coordinate work among programmers*
- *The DevOps projects use build tools (such as Gradle or Maven) to automate builds allowing for automatic download and configuration of dependencies or other libraries*
- *The DevOps projects use testing tools such as JUnit or Selenium which provides a framework for testing applications.*
- *The DevOps projects use deployment tools (such as Docker) which delivered software in packages called containers*

These items indicate that planning, CI/CD, version control, build, testing and deployment technologies are necessary for DevOps success. Although Azad (2022) indicated that technologies are important for DevOps success, this conclusion was achieved using qualitative findings from the literature. The current study established a significant positive relationship using inferential statistics. Furthermore, the beta coefficient of 0.289 indicates that this is the second most critical factor for DevOps success.

Culture

A positive significant relationship between technology and success was established using SEM. This signified that culture positively influenced DevOps success. The items that played a role in achieving this significance were:

- *DevOps team have a culture that espouses a shared understanding between key role players (such as developers, quality assurance and operations) that ensure a sharing of responsibility for the software they build.*
- *DevOps team have a strong focus on continuous improvement to optimise performance, cost and speed of delivery.*
- *DevOps team has a culture of embracing failures so they can turn failures into opportunities to learn.*

These items indicate that a culture of shared understanding, continuous improvement, and embracing failures are necessary for DevOps' success. In studies that were conducted using a qualitative orientation, Rowse and Cohen (2021) as well as Lwakatare et al. (2016) suggested that culture is a crucial factor that influences DevOps success. These assertions are corroborated by the quantitative evidence and the outcome of SEM exercise. Furthermore, the beta coefficient of 0.208 in this study indicates that this is the 3rd most important critical factor for DevOps success.

Project factors

A positive significant relationship between project factors and success was established using SEM. This signified that project factors positively influenced DevOps success. The items that played a role in achieving this significance were:

- *The DevOps projects has up-front risk analysis done.*
- *The DevOps projects are characterised as business mission-critical software.*

These items indicate that the project factors of upfront risk analysis and business mission-critical software are necessary for DevOps success. Furthermore, this study's beta coefficient of 0.189 indicates that this is the fourth most important critical factor for DevOps success.

8.4.2 Influence of Acceptance Factors on Behavioural Intention

Results from this study indicate that performance expectancy (PE), effort expectancy (EE), and social influence (SI) significantly influenced BI (BI).

Performance Expectancy

SEM results indicated a positive significant relationship between PE and BI. Hence, PE positively influenced BI. Subsequent to the EFA and CFA exercise, the items that were significant in achieving this result were:

- *I find DevOps useful for software development and deployment.*
- *Using DevOps increases my chances of achieving things that are important during software development and deployment.*
- *Using DevOps helps me accomplish software development and deployment tasks quickly.*
- *Using DevOps increases my productivity within a DevOps team.*

These items indicate that when DevOps is useful for development and deployment thereby increasing productivity there is an increased BI to use DevOps. Karunarathna et al. (2023) found that PE positively influences BI to use DevOps. The beta coefficient of 0.462 indicates that this is the most significant factor that influences the BI to use DevOps.

Effort Expectancy

SEM results indicated a positive significant relationship between EE and BI. Hence, EE positively influenced BI. Subsequent to the EFA and CFA exercise, the items that were significant in achieving this result were:

- *Learning how to use DevOps framework is easy for me.*
- *My interaction with DevOps is clear and understandable.*
- *It is easy for me to become skillful in using DevOps.*

These items indicate that when professionals find it easy to become skillful, their interactions are clear and understandable and it is easy for them to learn the framework, thus there is an increased BI to use DevOps. This finding is in keeping with Huq (2017) and Chiyangwa (2017) demonstrated that EE positively influenced BI in an Agile environment. The beta coefficient of 0.289 indicates that this is the second critical factor influencing BI.

Social Influence

SEM results indicated a positive significant relationship between SI and BI. Hence, SI positively influenced BI. Subsequent to the EFA and CFA exercise, the items that were significant in achieving this result were:

- *People who are important to me think that I should use DevOps for software development*
- *People who influence my behaviour think that I should use DevOps for software development.*
  .
- *People whose opinions that I value prefer that I use DevOps for software development*

These items indicate that when professionals are influenced by other people's opinions, that is by people who are likely to influence their behaviour and whose opinions they value, there is an increased BI to use DevOps. Kipreos (2019) demonstrated that EE positively influenced BI in an Agile environment. The beta coefficient of 0.174 indicates that this is the 3rd most critical factor that influences the BI to use DevOps.

8.4.3 Factors Influencing Actual Usage

Results from this study indicate that actual usage is influenced by behavioural intention (BI) and technology.

Behavioural Intention

SEM results indicated a positive significant relationship between BI and Actual Usage. Hence, BI positively influenced actual usage. Subsequent to the EFA and CFA exercise, the items that were significant in achieving this result were:

- *I intend to use a DevOps approach for software development projects on a regular basis.*
- *I intend to make an effort to practice a DevOps approach for software development projects on a regular basis.*
- *I predict that I will be able to use a DevOps approach for software development projects in my organisation*
- *Based on my knowledge of DevOps methodology, I predict that I would make an effort to use it more often*

These items indicate that when professionals intend to use DevOps regularly or predict to use it, then there is an increase in actual usage of DevOps. Mudarikwa and Grace (2018) demonstrated that BI positively influenced actual usage in an Agile environment. The beta coefficient of 0.223 indicates that BI is important in influencing actual usage.

Tools and Technologies

Results suggested a new positive significant relationship between technology and actual usage. Hence, technology positively influenced actual usage. Subsequent to the EFA and CFA exercise, the items that were significant in achieving this result were:

- *The DevOps projects use planning tools (such as Jira) for planning and project tracking allowing teams to ship code early and often.*
- *The DevOps projects use continuous integration and continuous delivery tools (such as Jenkins)*
- *The DevOps projects use Version Control tools (such as Git and GitHub) to coordinate work among programmers*
- *The DevOps projects use build tools (such as Gradle or Maven) to automate builds allowing for automatic download and configuration of dependencies or other libraries*
- *The DevOps projects use testing tools such as JUnit or Selenium which provides a framework for testing applications.*
- *The DevOps projects use deployment tools (such as Docker) which delivered software in packages called containers*

These items indicate that planning, CI/CD, version control, build, testing and deployment technologies lead to increased actual usage. The beta coefficient of 0.244 indicates that technology is important factor in increasing actual usage.

8.4.4 Factors that Influence Performance Expectancy

Results suggested that performance expectancy is influenced by automation, technology and hedonic motivation (HM).

Automation

A positive significant relationship between automation and PE was established using SEM. Therefore, automation positively influenced PE. Subsequent to the EFA and CFA exercise, the items that were significant in achieving this result were:

- *DevOps projects use automated execution for continuous testing*
- *DevOps projects use an automated build trigger for continuous integration*
- *DevOps projects use automated performance and error notifications during continuous delivery*

These items indicate that automated execution of testing, automated build trigger, automated performance, and automated error notifications increase PE. The beta coefficient of 0.307 indicates that automation is important in increasing PE in DevOps.

Tools and Technologies

SEM results indicated a positive significant relationship between technology and PE. Hence, technology positively influenced PE. Subsequent to the EFA and CFA exercise, the items that were significant in achieving this result were:

- *The DevOps projects use planning tools (such as Jira) for planning and project tracking allowing teams to ship code early and often.*
- *The DevOps projects use continuous integration and continuous delivery tools (such as Jenkins)*
- *The DevOps projects use Version Control tools (such as Git and GitHub) to coordinate work among programmers*
- *The DevOps projects use build tools (such as Gradle or Maven) to automate builds allowing for automatic download and configuration of dependencies or other libraries*
- *The DevOps projects use testing tools such as JUnit or Selenium which provides a framework for testing applications.*
- *The DevOps projects use deployment tools (such as Docker) which delivered software in packages called containers*

These items indicate that planning, CI/CD, version control, build, testing and deployment technologies are important in increasing PE. Furthermore, the beta coefficient of 0.208 indicates that technology is essential in increasing PE.

Hedonic Motivation

A positive significant relationship between hedonic motivation (HM) and PE was established using SEM. Therefore, HM positively influenced PE. The items after EFA and CFA that were significant in achieving this result were:

- *I enjoy interaction with team members in other domains.*
- *I am motivated to work since things get done faster and problem solved more efficiently and effectively.*
- *I have fun playing with and investigating new tools to make the DevOps strategy better.*

These items indicate that interacting with team members in other domains, getting things done faster and better and having fun investigating new tools increase PE. The beta coefficient of 0.414 indicates that HM is essential in increasing PE. HM's influence on PE is not widely established, with Tamilmani, Rana, Prakasam, and Dwivedi (2019) highlighting two instances in the literature demonstrating a significant finding between HM and EE.

8.4.5 Factors that Influence Performance Expectancy

Results suggested that effort expectancy is influenced by technology and hedonic motivation (HM).

Tools and Technologies

SEM results indicated a positive significant relationship between technology and EE. Hence, technology positively influenced EE. Subsequent to the EFA and CFA exercise, the items that were significant in achieving this result were:

- *The DevOps projects use planning tools (such as Jira) for planning and project tracking allowing teams to ship code early and often.*
- *The DevOps projects use continuous integration and continuous delivery tools (such as Jenkins)*
- *The DevOps projects use Version Control tools (such as Git and GitHub) to coordinate work among programmers*
- *The DevOps projects use build tools (such as Gradle or Maven) to automate builds allowing for automatic download and configuration of dependencies or other libraries*
- *The DevOps projects use testing tools such as JUnit or Selenium which provides a framework for testing applications.*
- *The DevOps projects use deployment tools (such as Docker) which delivered software in packages called containers*

These items indicate that planning, CI/CD, version control, build, testing and deployment technologies are important in increasing EE. Furthermore, the beta coefficient of 0.271 indicates that technology is essential to increasing EE.

Hedonic Motivation

A positive significant relationship between hedonic motivation (HM) and EE was established using SEM. Therefore, HM positively influenced EE. Subsequent to the EFA and CFA exercise, the items that were significant in achieving this result were:

- *I enjoy interaction with team members in other domains.*
- *I am motivated to work since things get done faster and problem solved more efficiently and effectively.*
- *I have fun playing with and investigating new tools to make the DevOps strategy better.*

These items indicate that interacting with team members in other domains, getting things done faster and better and having fun investigating new tools increase the ease with which to implement DevOps, thus supporting the positive relationship between HM and EE. values recorded for EE. The beta coefficient of 0.525 indicates that HM is important in increasing EE. The implication here is that the greater the HM, the less the effort it takes to make use of DevOps. The HM's influence on EE is not

widely established, with Tamilmani et al. (2019) highlighting three instances in the literature that demonstrated a significant finding between HM and EE.

8.5 DEVOPS CRITICAL SUCCESS AND ACCEPTANCE MODEL

Figure 8.1 shows the final model obtained after Chapter 7 data analysis. It is evident from this model that automaton, tools and technology, culture and project factors were the critical factors influencing DevOps success. Performance expectancy, effort expectancy and social influence were critical factors influencing BI. BI was found to influence the Actual Usage of DevOps significantly. Of the additional four paths established in Chapter 5, only three were significant (Technology → PE, Technology → EE and Automation → PE). SEM also suggested three significant paths (Technology → Actual Usage of DevOps, HM → PE and HM → EE). The beta coefficients indicated in the figure illustrate the level of influence each factor has on the dependent factors (Actual Usage, Success, BI, PE, EE).

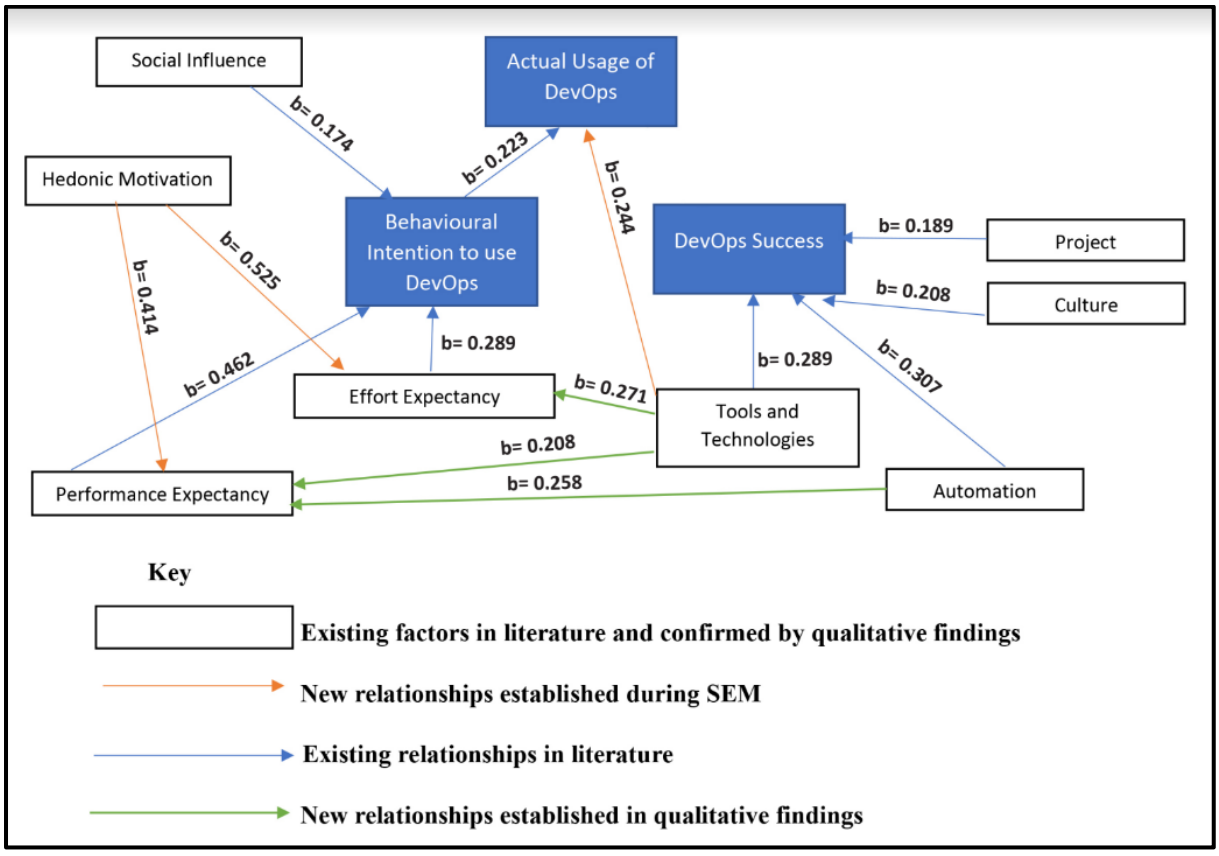

Figure 8.1: DevOps critical success and acceptance model

8.6 SUMMARY

This chapter provided a detailed discussion of the results reported Chapters 5 (qualitative) and Chapter 7 (quantitative). Where appropriate, the triangulation of the qualitative and quantitative results was explained. Furthermore, the results of this study were compared with those of the previous studies. The next chapter will be the concluding chapter, which will summarize all the chapters and discuss the study's contribution to theory and practice. It will conclude by identifying the limitations of the current study and the potential for future work.

CHAPTER 9: CONCLUSION, RESEARCH CONTRIBUTIONS AND FUTURE WORK

9.1 INTRODUCTION

The previous chapter provided an in-depth discussion of the results achieved in this study culminating in the DevOps acceptance and success model. This chapter provides a general overview of the study from inception to conclusion, showing how this study contributes to theory and practice. Potential limitations and recommendations for future research are discussed.

9.2 RESEARCH OVERVIEW

The current section provides a concise summary of the 8 primary chapters that comprises this thesis and the efforts that were made to accomplish the study's goals and objectives.

- **Chapter 1 – Introduction to the study**

This chapter presented an introduction to software development and its importance to organisations. It elaborated on the current Agile approach and its shortcomings, resulting in the need for DevOps. The problem statement highlighted the lack of research on DevOps acceptance and success factors. Research questions and objectives were generated based on this problem.

- **Chapter 2 – The past, present and future of software development processes**

This chapter comprehensively discussed the evolution of software process models and their shortcomings. It emphasised how the process models are not conducive to the modern business environments where there is a critical need to develop and deliver software systems quickly. Agile is discussed as a solution to this problem. Although Agile creates software systems quickly, it is still not quickly released into the operational environment. DevOps is discussed as a model that enables quick deployment of software. The chapter ends by explaining DevOp's role in creating software factories and the 4th Industrial Revolution.

- **Chapter 3 - Theoretical underpinning and the study's preliminary conceptual model**

This chapter was divided into 3 main sections. The 1st section discussed the success factors affecting the software development success. A majority of the studies that were analysed were from an Agile perspective. These studies were classified as relatively current and it was observed that each of these studies made a reference to DevOps. This observation necessitated an incursion into the domain of success factors that influence DevOps. The 2nd part of this chapter was devoted to an evaluation of the acceptance models pertaining to technology adoption, culminating in a discussion of the merits of resorting to the UTAUT 2 model as an ideal theoretical foundation for the study. The final part of

the chapter incorporated the acceptance and success factors review by generating a preliminary conceptual model.

- **Chapter 4 - Research Design and Methodology**

This chapter presented various research designs, of which the pragmatic approach was deemed to be the most appropriate for this study because it allowed for both inductive and deductive approaches to be used. Therefore, this study used a sequential research method, with a qualitative phase that was driven by a data collection strategy that entailed semi-structured interviews. The data analysis was conducted using content analysis with the aid of the qualitative data analysis tool named Nvivo. The quantitative part of the study used the survey technique in the form of a questionnaire to collect data. The data was analysed using descriptive and inferential statistics and aided by the quantitative data analysis tools named IBM SPSS and AMOS.

- **Chapter 5 - Qualitative analysis of the lived experience of DevOps professionals**

The semi-structured interview data was analysed in this chapter using content analysis. Various themes were generated to answer the research questions 1 and 2. The main factors affecting success and acceptance identified in Chapter 3 were confirmed with empirical qualitative evidence. Furthermore, two additional factors (cloud and security) were identified with four additional paths that contributed to the study's conceptual model.

- **Chapter 6 - Revised conceptual model and study's hypotheses**

The qualitative findings were used to revise the preliminary conceptual model and generate a set of research hypotheses. The research hypotheses were validated by aligning the hypotheses to current literature as well as evidence provided in the qualitative data collection phase of the study.

- **Chapter 7 – Quantitative data analysis**

The quantitative data collected via questionnaires were analysed in this chapter. Results were presented using descriptive and inferential statistics. Most respondents agreed with the acceptance and success constructs investigated. Structural Equation Modelling (SEM) identified the main success factors as automation, technology, culture and project factors. The acceptance factors were identified as performance expectancy, effort expectancy and social influence.

- **Chapter 8 - Discussion of findings**

A discussion of the findings was presented in this chapter and contextualised to the research questions of this study. The triangulation of the qualitative and quantitative findings was explained and the significant paths of the model were discussed. A DevOps success and acceptance model was also presented.

- **Chapter 9 – Conclusion, research contributions and future work**

This chapter will discuss how this study contributes to theory and practice. The study's limitations and future work is also presented.

9.3 CONTRIBUTION OF THE RESEARCH

9.3.1 Contribution to Theory

This paper makes several significant contributions to IS development, especially DevOps. The most significant outcome of this research was to model the DevOps critical success and acceptance factors. The analysis of the relevant literature has provided evidence that there is a dearth of research on DevOps acceptance and success factors. The contribution of this study to theory is as follows:

1) The initial literature provided a preliminary conceptual model integrating the success and acceptance factors into a single model. This study combines the technology acceptance models and success theory into a single model, within the DevOps context. This model can serve as a springboard for other researchers who study acceptance and success factors in the context of software development methodology.
2) This study measured acceptance and success factors using a quantitative approach. Many of the previous studies were qualitative and did not significantly demonstrate the inter relationship between variables as well as the dependent variables' impact on the independent variables. This is evident from the discussion section where it is stipulated that the generalisations emanating from the current study could not be compared to previous studies because of a dearth of studies that have approached this topic comprehensively.
3) During the qualitative analysis, this study established 4 new paths in the conceptual model, of which 3 were statistically significant. This adds to the theory of significant factors (technology and automation) that determine PE and EE. Furthermore, 2 additional paths were derived through SEM that established a relationship between Hedonic Motivation, PE and EE. This supports the limited existing theory that HM influences PE and EE. These new relationships do present implications for the UTAUT-2 model because the relationship between HM, PE and EE could become a firmly entrenched component of this model. This assertion however is subject to further inquiry of this topic in an organisational setting.

9.3.2 Contribution to Practice

Apart from the theoretical contributions highlighted in the previous section, this study contributes to professional practice. This study established a critical acceptance and success model that can be used to guide DevOps implementation and adoption in the industrial sector. One of the significant outcomes that was illustrated in the study's final conceptual model was that the 3 most CSFs important for DevOps' success are automation, technology, and culture. Hence, organisations that

want to follow the DevOps route must incorporate automation and technology into their DevOps pipeline. Furthermore, a culture of learning, accepting shared responsibility and, where possible, automating their processes must be embedded within project teams. This study also identified performance expectancy, social influence and effort expectancy as factors that might increase usage. Hence, by promoting the right tools and technologies, the DevOps effort and level of complexity will be significantly reduced while the level of productivity will be increased. Hedonic Motivation also plays an important role in influencing performance expectancy and effort expectancy, so organisations are compelled to create a DevOps environment that is user friendly and a comfortable space for the development and operations teams to work in.

9.4 LIMITATIONS OF THE STUDY

1. Due to the difficulty of generating an accurate sampling frame, a non-probability sampling method was used during the quantitative phase. From a statistical perspective, non-probability sampling does have an impeding influence on any attempt to generalise the outcome from the study. This setback has been obviated to a certain extent by the study's comprehensive sample, however the lack of random sampling does compromise the generalisability of the results.
2. Although the DevOps acceptance and success model provided valuable insights into the factors affecting DevOps adoption, it may be difficult to use this model for other countries because of the cultural differences and varying resource allocation within countries.
3. Although the researcher practised various validity principles to ensure that the research instrument accurately captured the constructs under investigation, some constructs were found to be unreliable measures and were removed from analysis, thereby preventing a more comprehensive model from being created.
4. The majority of participants in this study were male. This may have introduced a potential gender bias in the findings, thus affecting the generalisability of the results.

9.5 FUTURE WORK

1. The qualitative results alluded to age, personality, and experience as pivotal factors that play a role in the acceptance of DevOps. Hence, a similar study can be carried out to test these mediation factors on success and acceptance.
2. This study aggregated responses from various stakeholders. A stratified sampling approach can be used to determine factors affecting different stakeholder groups (such as business managers, systems analysts, software developers and software testers, project managers and

Scrum masters as well as operations engineers). This can be used to inform implementors how the acceptance and success factors might affect various stakeholders differently, thus allowing organisations to implement different strategies to promote DevOps in the organisation.

3. The squared regression weights indicate that for both success and acceptance factors, other elements can still affect the dependent variables. This knowledge could be supplemented in future studies where the focus could be on identifying more factors to add to the model and improve the research instrument.

9.6 SUMMARY

In conclusion, the rapid growth of the Internet and digital technologies have transformed society, making organisations more reliant on software. This has resulted in the need for continuous deployment of software. To achieve this, a DevOps paradigm needs to be implemented. However, for such a paradigm to be successful and accepted by software professionals, various factors influencing success and acceptance must be identified to inform the stakeholders of the need to create a conducive environment to facilitate DevOps acceptance and success. While these factors may be accidentally discovered via anecodotal experiences and evidence, the current study provides solid empirical evidence to support the acceptance and success factors that have been identified. This was achieved by invoking software development methodological theory, academic literature on the topic in conjunction with a mixed methods research approach that provided the researcher with an opportunity to investigate the phenomenon from a depth (qualitative) perspective and at scale (quantitative). The thesis has achieved the objectives that have been set out at the inception of the study. Finally, the study's limitations and prospect for future research have been presented.

REFERENCES

Abbas, N., Gravell, A. M., & Wills, G. B. (2008). Historical roots of Agile methods: where did "Agile thinking" come from? In *Agile Processes in Software Engineering and Extreme Programming* (pp. 94-103). Springer.

Abd, T., Abd, M., & Kholief, S. (2016). Identify and Classify Critical Success Factor of Agile Software Development Methodology Using Mind Map. *International Journal of Advanced Computer Science and Applications*, *7*. https://doi.org/10.14569/IJACSA.2016.070513

Abdi, H. (2003). Factor rotations in factor analyses. *Encyclopedia for Research Methods for the Social Sciences. Sage: Thousand Oaks, CA*, 792-795.

Abiola, O. B., & Olufemi, O. G. (2023). An Enhanced CICD Pipeline: A DevSecOps Approach. *International Journal of Computer Applications*, *975*, 8887.

Aboelmaged, M., & Gebba, T. (2013). Mobile Banking Adoption: An Examination of Technology Acceptance Model and Theory of Planned Behavior. *International Journal of Business Research and Development*, *2*, 35-50. https://doi.org/10.24102/ijbrd.v2i1.263

Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2017). Agile software development methods: Review and analysis. *arXiv preprint arXiv:1709.08439*.

Abubakar, F. M., & Ahmad, H. B. (2013). The moderating effect of technology awareness on the relationship between UTAUT constructs and behavioural intention to use technology: A conceptual paper. *Australian journal of business and management research*, *3*(2), 14-23.

Achar, S. (2021). Enterprise SaaS Workloads on New-Generation Infrastructure-as-Code (IaC) on Multi-Cloud Platforms. *Global Disclosure of Economics and Business*, *10*(2), 55-74.

Adams, D. A., Nelson, R. R., & Todd, P. A. (1992). Perceived usefulness, ease of use, and usage of information technology: A replication. *MIS Quarterly*, 227-247.

Adeoye-Olatunde, O. A., & Olenik, N. L. (2021). Research and scholarly methods: Semi-structured interviews. *Journal of the american college of clinical pharmacy*, *4*(10), 1358-1367.

Agarwal, A., Gupta, S., & Choudhury, T. (2018). Continuous and integrated software development using DevOps. 2018 International conference on advances in computing and communication engineering (ICACCE),

Agarwal, R., & Prasad, J. (1998). The antecedents and consequents of user perceptions in information technology adoption. *Decision Support Systems*, *22*(1), 15-29. https://doi.org/http://dx.doi.org/10.1016/S0167-9236(97)00006-7

Ågerfalk, P. J., Fitzgerald, B., & Slaughter, S. A. (2009). Flexible and distributed information systems development: State of the art and research challenges. *Information Systems Research*.

Ahmad, S., Bhatti, S. H., & Hwang, Y. (2020). E-service quality and actual use of e-banking: Explanation through the Technology Acceptance Model. *Information Development*, *36*(4), 503-519.

Ahmed, F., Bottacci, E., Rantanen, S., Romppanen, J., & Verschuren, N. (2021). Devops Practices For Software Development And Consulting Firms. *Lut School Of Engineering Science*.

Ahonen, J. (2020). Management and Development of DevOps and Traditional IT Project Paths.

Aiello, B., & Sachs, L. (2016). *Agile Application Lifecycle Management: Using DevOps to Drive Process Improvement*. Addison-Wesley Professional.

Ajiga, D., Okeleke, P. A., Folorunsho, S. O., & Ezeigweneme, C. (2024). Methodologies for developing scalable software frameworks that support growing business needs.

Ajzen, I. (1985). From intentions to actions: A theory of planned behavior. In *Action control* (pp. 11-39). Springer.

Ajzen, I. (1991). The theory of planned behavior. *Organizational behavior and human decision processes*, *50*(2), 179-211.

Ajzen, I., & Fishbein, M. (1969). The prediction of behavioral intentions in a choice situation. *Journal of Experimental Social Psychology*, *5*(4), 400-416. https://doi.org/10.1016/0022-1031(69)90033-X

Akbar, M. A., Mahmood, S., Shafiq, M., Alsanad, A., Alsanad, A. A.-A., & Gumaei, A. (2020). Identification and prioritization of DevOps success factors using fuzzy-AHP approach. *Soft computing*, 1-25.

Akbar, M. A., Rafi, S., Alsanad, A. A., Qadri, S. F., Alsanad, A., & Alothaim, A. (2022). Toward Successful DevOps: A Decision-Making Framework. *IEEE Access*, *10*, 51343-51362.

Akman, I., & Mishra, A. (2014). Green Information Technology Practices among IT Professionals: Theory of Planned Behavior Perspective. *Problemy Ekorozwoju*, *9*, 47-54.

Akshaya, H., Vidya, J., & Veena, K. (2015). A basic introduction to devops tools. *International Journal of Computer Science & Information Technologies*, *6*(3), 05-06.

Al-Jabri, I. M., & Sohail, M. S. (2012). Mobile banking adoption: Application of diffusion of innovation theory. *Journal of Electronic Commerce Research*, *13*(4), 379-391.

Al-Mamary, Y. H. S. (2022). Understanding the use of learning management systems by undergraduate university students using the UTAUT model: Credible evidence from Saudi Arabia. *International Journal of Information Management Data Insights*, *2*(2), 100092.

Al-Zewairi, M., Biltawi, M., Etaiwi, W., & Shaout, A. (2017). Agile Software Development Methodologies: Survey of Surveys. *Journal of Computer and Communications*, *Vol.05No.05*, 24, Article 75114. https://doi.org/10.4236/jcc.2017.55007

Al Masud, S. M. R., Masnun, M., Sultana, A., Sultana, A., Ahmed, F., & Begum, N. (2022). DevOps Enabled Agile: Combining Agile and DevOps Methodologies for Software Development. *International Journal of Advanced Computer Science and Applications*, *13*(11).

Alavi, M. (1984). An assessment of the prototyping approach to information systems development. *Commun. ACM*, *27*(6), 556–563. https://doi.org/10.1145/358080.358095

Aldahmash, A., Gravell, A. M., & Howard, Y. (2017). A Review on the Critical Success Factors of Agile Software Development. In J. Stolfa, S. Stolfa, R. V. O'Connor, & R. Messnarz, *Systems, Software and Services Process Improvement* Cham.

Alfadda, H. A., & Mahdi, H. S. (2021). Measuring students' use of zoom application in language course based on the technology acceptance model (TAM). *Journal of Psycholinguistic Research*, *50*(4), 883-900.

Algaith, A., Nunes, P., Jose, F., Gashi, I., & Vieira, M. (2018). Finding SQL injection and cross site scripting vulnerabilities with diverse static analysis tools. 2018 14th European dependable computing conference (EDCC),

Alhajeri, A. A. A. M. A., & Safian, E. E. M. (2023). Mediating Role of Organizational Learning in the Relationship between Use of Artificial Intelligence Security Technology and Community Security. *International Journal of Sustainable Construction Engineering and Technology*, *14*(3), 137-153.

Alhazmi, A. A., & Kaufmann, A. (2022). Phenomenological qualitative methods applied to the analysis of cross-cultural experience in novel educational social contexts. *Frontiers in psychology*, *13*, 1495.

Alkhowaiter, W. A. (2022). Use and behavioural intention of m-payment in GCC countries: Extending meta-UTAUT with trust and Islamic religiosity. *Journal of Innovation & Knowledge*, *7*(4), 100240.

Almeida, F., Simões, J., & Lopes, S. (2022). Exploring the Benefits of Combining DevOps and Agile. *Future Internet*, *14*(2), 63. https://www.mdpi.com/1999-5903/14/2/63

Alomari, A. S., & Abdullah, N. L. (2023). Factors influencing the behavioral intention to use Cryptocurrency among Saudi Arabian public university students: Moderating role of financial literacy. *Cogent Business & Management*, *10*(1), 2178092.

Alshammari, S. H., & Rosli, M. S. (2020). A review of technology acceptance models and theories. *Innovative Teaching and Learning Journal (ITLJ)*, *4*(2), 12-22.

Alshamrani, A., & Bahattab, A. (2015). A Comparison Between Three SDLC Models Waterfall Model, Spiral Model, and Incremental/Iterative Model. *International Journal of Computer Science Issues (IJCSI)*, *12*(1), 106-111. https://ukzn.idm.oclc.org/login?url=https://www.proquest.com/scholarly-

journals/comparison-between-three-sdlc-models-waterfall/docview/1660801422/se-2?accountid=158225
Alves, L. M. A. (2022). *Software defined applications: a DevOps approach to monitoring* 2022.
Ambler, S. (2005). A manager's introduction to the Rational Unified Process (RUP). *Version: December*, *4*, 2005.
Ambler, S. (2012). Agile Adoption Strategies: November 2011 Survey Results. https://ambysoft.com/surveys/agilestateofart201111.html
Analysis_Inn. (2020). *How to calculate Average Variance Extracted and Composite Reliability*. Retrieved 20 April 2023 from https://www.analysisinn.com/post/how-to-calculate-average-variance-extracted-and-composite-reliability/
Anderson, A. J. (2019). *Examination of Adoption Theory on the DevOps Practice of Continuous Delivery*
Angara, J., Prasad, S., & Sridevi, G. (2020). DevOPs project management tools for sprint planning, estimation and execution maturity. *Cybernetics and Information Technologies*, *20*(2), 79-92.
Angara, S. V. J. (2020). AN EXTENSIVE RISK-MITIGATING FRAMEWORK FOR CONTINUOUS TESTING USING DEVOPs.
Ariza-Montes, A., Quan, W., Radic, A., Koo, B., Kim, J. J., Chua, B.-L., & Han, H. (2023). Understanding the behavioral intention to use urban air autonomous vehicles. *Technological Forecasting and Social Change*, *191*, 122483. https://doi.org/https://doi.org/10.1016/j.techfore.2023.122483
Armitage, C. J., & Conner, M. (2001). Efficacy of the Theory of Planned Behaviour: a meta-analytic review. *Br J Soc Psychol*, *40*(Pt 4), 471-499. https://doi.org/10.1348/014466601164939
Artac, M., Borovssak, T., Di Nitto, E., Guerriero, M., & Tamburri, D. A. (2017). DevOps: introducing infrastructure-as-code. 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C),
Arulkumar, V., & Lathamanju, R. (2019). Start to finish automation achieve on cloud with build channel: By DevOps method. *Procedia Computer Science*, *165*, 399-405.
Ashenden, D., & Ollis, G. (2020). Putting the sec in devsecops: Using social practice theory to improve secure software development. New Security Paradigms Workshop 2020,
Augustine, S., Payne, B., Sencindiver, F., & Woodcock, S. (2005). Agile project management: steering from the edges. *Communications of the ACM*, *48*(12), 85-89.
Avison, D., & Fitzgerald, G. (2006). Methodologies for Developing Information Systems: A Historical Perspective. In (Vol. 214, pp. 27-38). https://doi.org/10.1007/978-0-387-34732-5_3
Awad, M. (2005). A comparison between agile and traditional software development methodologies. *University of Western Australia*, *30*, 1-69.
Azad, N. (2022). Understanding DevOps critical success factors and organizational practices. 2022 IEEE/ACM International Workshop on Software-Intensive Business (IWSiB),
Babbie, E., & Mouton, J. (2008). *The practice of social research (South African ed.)*. Oxford University Press Southern Africa.
Baccarini, D. (1999). The Logical Framework Method for Defining Project Success. *Project Management Journal*, *Volume 30*, 25-32. https://doi.org/10.1177/875697289903000405
Bagozzi, R. P. (2007). The legacy of the technology acceptance model and a proposal for a paradigm shift. *Journal of the association for information systems*, *8*(4), 3.
Bahadori, K., & Vardanega, T. (2019). Devops meets dynamic orchestration. Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment: First International Workshop, DEVOPS 2018, Chateau de Villebrumier, France, March 5-6, 2018, Revised Selected Papers 1,
Balachandran, D., Kovas, M., Kumaran, V., & Dawson, Z. (2016). Launching Agile and DevOps in a 100 Year Old Company.
Bandura, A. (1977). Self-efficacy: Toward a unifying theory of behavioral change. *Psychological Review*, *84*(2), 191-215. https://doi.org/10.1037/0033-295X.84.2.191

Bandura, A. (1986a). The explanatory and predictive scope of self-efficacy theory. *Journal of social and clinical psychology*, *4*(3), 359-373.
Bandura, A. (1986b). *Social foundations of thought and action: A social cognitive theory*. Prentice-Hall, Inc.
Bandura, A. (1999). Social Cognitive Theory: An Agentic Perspective. *Asian Journal of Social Psychology*, *2*(1), 21-41. https://doi.org/https://doi.org/10.1111/1467-839X.00024
Bandura, A. (2001). Social cognitive theory: an agentic perspective. *Annual review of psychology*, *52*, 1-26. https://doi.org/10.1146/annurev.psych.52.1.1
Barrett, P. (2007). Structural equation modelling: Adjudging model fit. *Personality and Individual Differences*, *42*(5), 815-824. https://doi.org/https://doi.org/10.1016/j.paid.2006.09.018
Basili, V. R., & Turner, A. J. (1975). Iterative enhancement: A practical technique for software development. *IEEE Transactions on Software Engineering*, *1*(4), 390-396.
Bass, L., Weber, I., & Zhu, L. (2015). *DevOps: A Software Architect's Perspective*. Addison-Wesley Professional.
Batra, P., & Jatain, A. (2020). Measurement Based Performance Evaluation of DevOps. 2020 International Conference on Computational Performance Evaluation (ComPE),
Battina, D. S. (2020). Devops, A New Approach To Cloud Development & Testing. *International Journal of Emerging Technologies and Innovative Research (www. jetir. org), ISSN*, 2349-5162.
Beck, K. (1999). Embracing change with extreme programming. *Computer*, *32*(10), 70-77.
Beck, K. (2000). *Extreme programming explained: embrace change*. Addison-Wesley Professional.
Beck, K., Beedle, M., Bennekum, A. V., Cockburn, A., Cunningham, W., Fowler, M., . . . Jeffries, R. (2001). *Agile manifesto*. http://agilemanifesto.org/
Bell, T. E., & Thayer, T. A. (1976). *Software requirements: Are they really a problem?* Proceedings of the 2nd international conference on Software engineering, San Francisco, California, USA.
Benington, H. D. (1983). Production of Large Computer Programs. *Annals of the History of Computing*, *5*(4), 350-361. https://doi.org/10.1109/MAHC.1983.10102
Bentler, P., & Chou, C.-P. (1987). Practical Issues in Structural Equation Modeling. *Sociological Methods & Research*, *16*. https://doi.org/10.1177/0049124187016001004
Bentler, P. M. (2006). EQS 6 : structural equations program manual.
Bentler, P. M., & Bonett, D. G. (1980). Significance tests and goodness of fit in the analysis of covariance structures. *Psychological Bulletin*, *88*(3), 588-606. https://doi.org/10.1037/0033-2909.88.3.588
Bhuvaneswari, T., & Prabaharan, S. (2013). A survey on software development life cycle models. *International Journal of Computer Science and Mobile Computing*, *2*(5), 262-267.
Bigham, T., & Litchfield, E. (2022). DevRiskOps: Risk management in DevOps. https://www2.deloitte.com/uk/en/blog/risk-powers-performance/2022/risk-management-in-devops.html
Blumberg, B. F., Cooper, D. R., & Schindler, P. S. (2005). Survey research. *Business research methods*(243-276).
Boehm, B. (1988). A spiral model of software development and enhancement. *Computer*, *21*(5), 61-72.
Boehm, B., Gray, T. E., & Seewaldt, T. (1984). *Prototyping vs. specifying: A multi-project experiment* Proceedings of the 7th international conference on Software engineering, Orlando, Florida, USA.
Boehm, B., & Hansen, W. J. (2000). *Spiral development: Experience, principles, and refinements*.
Boehm, B., & Turner, R. (2003). People factors in software management: lessons from comparing agile and plan-driven methods. *CrossTalk: The Journal of Defense Software Engineering*, *16*(12), 4-8.
Bogopa, M. E., & Marnewick, C. (2022). Critical success factors in Software Development Projects. *South African Computer Journal*, *34*(1).

Bollen, K. A., & Stine, R. A. (1992). Bootstrapping Goodness-of-Fit Measures in Structural Equation Models. *Sociological Methods & Research*, *21*(2), 205-229. https://doi.org/10.1177/0049124192021002004

Bolt, M. A., Killough, L. N., & Koh, H. C. (2001). Testing the Interaction Effects of Task Complexity in Computer Training Using the Social Cognitive Model. *Decision Sciences*, *32*(1), 1-20. https://doi.org/https://doi.org/10.1111/j.1540-5915.2001.tb00951.x

Bosch, J. (2014). Continuous Software Engineering: An Introduction. 3-13. https://doi.org/10.1007/978-3-319-11283-1-1

Bou Ghantous, G., & Gill, A. (2017). DevOps: Concepts, practices, tools, benefits and challenges. *PACIS2017*.

Boynton, A. C., & Zmud, R. W. (1984). An assessment of critical success factors. *Sloan management review*, *25*(4), 17-27.

Bradley, T. (2015). *DevOps is natural evolution of Agile development*. Retrieved 20 March 2021 from https://techspective.net/2015/03/04/devops-is-natural-evolution-of-agile-development/#google_vignette

Brown, S., & Venkatesh, V. (2005). Model of Adoption of Technology in Households: A Baseline Model Test and Extension Incorporating Household Life Cycle. *MIS Quarterly*, *29*, 399-436. https://doi.org/10.2307/25148690

Browne, M. W. (1984). Asymptotically distribution-free methods for the analysis of covariance structures. *British Journal of Mathematical and Statistical Psychology*, *37*(1), 62-83. https://doi.org/https://doi.org/10.1111/j.2044-8317.1984.tb00789.x

Brunnert, A., van Hoorn, A., Willnecker, F., Danciu, A., Hasselbring, W., Heger, C., . . . von Kistowski, J. (2015). Performance-oriented DevOps: A research agenda. *arXiv preprint arXiv:1508.04752*.

Bucena, I., & Kirikova, M. (2017). Simplifying the DevOps Adoption Process. BIR Workshops,

Bullen, C., & Rockart, J. (1981). A primer on critical success factors.

Burgess-Allen, J., & Owen-Smith, V. (2010). Using mind mapping techniques for rapid qualitative data analysis in public participation processes. *Health Expectations*, *13*(4), 406-415.

Buttar, A. M., Khalid, A., Alenezi, M., Akbar, M. A., Rafi, S., Gumaei, A. H., & Riaz, M. T. (2023). Optimization of DevOps Transformation for Cloud-Based Applications. *Electronics*, *12*(2), 357.

Byrne, B. M. (2010). Structural Equation Modeling With AMOS: Basic Concepts, Applications, and Programming, [https://doi.org/10.4324/9780203805534]. *Second Edition (2nd ed.). Routledge.*

CA_Technologies. (2015). *Global study by CA Technologies finds DevOps a key contributor to the bottom line*. Retrieved 20 June 2020 from https://www.itweb.co.za/article/global-study-by-ca-technologies-finds-devops-a-key-contributor-to-the-bottom-line/XnWJadMbrY2MbjO1

Candy, P. C. (1989). Constructivism and the Study of Self-direction in Adult Learning. *Studies in the Education of Adults*, *21*(2), 95-116. https://doi.org/10.1080/02660830.1989.11730524

Cao, J., & Zhang, S. (2016). ITIL Incident management process reengineering in industry 4.0 environments. Proceedings of the 2nd International Conference on Advances in Mechanical Engineering and Industrial Informatics (AMEII 2016),

Capizzi, A., Distefano, S., & Mazzara, M. (2020, 2020//). From DevOps to DevDataOps: Data Management in DevOps Processes. Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment, Cham.

Carillo, K. D. (2010). Social cognitive theory in is research–literature review, criticism, and research agenda. International Conference on Information systems, Technology and management,

Carr, M., & Verner, J. (1997). Prototyping and software development approaches. *Department of Information Systems, City University of Hong Kong, Hong Kong*, 319-338.

Chandra, V. (2015). Comparison between Various Software Development Methodologies. *International Journal of Computer Applications*, *131*, 7-10. https://doi.org/10.5120/ijca2015907294

Charan, K., Karthikeya, N., Narasimharao, P. B., Devi, S. A., Abhilash, V., & Srinivas, P. (2023). Effective Code Testing Strategies in DevOps: A Comprehensive Study of Techniques and Tools for Ensuring Code Quality and Reliability. 2023 4th International Conference on Electronics and Sustainable Communication Systems (ICESC),

Chatterjee, S. (2022). Antecedents of Behavioral Intention Impacting Human Behavior to Use IoT-Enabled Devices: An Empirical Investigation. *International Journal of Technology and Human Interaction (IJTHI)*, *18*(1), 1-19.

Chauhan, M., & Shiaeles, S. (2023). An Analysis of Cloud Security Frameworks, Problems and Proposed Solutions. *Network*, *3*(3), 422-450. https://www.mdpi.com/2673-8732/3/3/18

Chawre, H. (2023). Enterprise DevOps: The Crucial Role of DevOps in Enterprise Application Development. https://www.turing.com/blog/enterprise-devops-benefits-tools-best-practices/

Checkmarx. (2020). *DevOps: An Integrated Approach to Embedding Security into DevOps - A Best Practices Guide*.

Chen, L. (2017). Continuous Delivery: Overcoming adoption challenges. *Journal of Systems and Software*, *128*, 72-86. https://doi.org/https://doi.org/10.1016/j.jss.2017.02.013

Chen, L., & Power, P. (2015). Continuous Delivery: Huge Benefits, but Challenges Too. *IEEE Software*, *32*(2), 50-54. https://doi.org/10.1109/MS.2015.27

Cheng, J. M., Kao, L. L., & Lin, J. Y.-C. (2004). An investigation of the diffusion of online games in Taiwan: An application of Rogers' diffusion of innovation theory. *Journal of American Academy of Business*, *5*(1/2), 439-445.

Cheon, J., Lee, S., Crooks, S. M., & Song, J. (2012). An investigation of mobile learning readiness in higher education based on the theory of planned behavior. *Computers & Education*, *59*(3), 1054-1064. https://doi.org/https://doi.org/10.1016/j.compedu.2012.04.015

Cheung, G. W., & Lau, R. S. (2008). Testing mediation and suppression effects of latent variables: Bootstrapping with structural equation models. *Organizational Research Methods*, *11*(2), 296-325.

Chiyangwa, T. B. (2017). *Modelling the critical success factors of agile software development projects in South Africa* University of South Africa, Pretoria]. http://hdl.handle.net/10500/24857

Chiyangwa, T. B., & Mnkandla, E. (2017). Modelling the critical success factors of agile software development projects in South Africa [traditional software development; vital; software engineering methodology; success factors; quantitative method]. *2017*, *19*(1). https://doi.org/10.4102/sajim.v19i1.838

Cho, J. (2008). Issues and Challenges of agile software development with SCRUM. *Issues in Information Systems*, *9*(2), 188-195.

Chow, T., & Cao, D.-B. (2008). A survey study of critical success factors in agile software projects. *Journal of Systems and Software*, *81*(6), 961-971. https://doi.org/http://dx.doi.org/10.1016/j.jss.2007.08.020

Chuttur, M. (2009). Overview of the technology acceptance model: Origins, developments and future directions. *Indiana University, USA. Sprouts: Working papers on Information Systems*, *9*.

Civelek, M. E. (2018). *Essentials of structural equation modeling*. Lulu. com.

Clarke, R. (1999). *A Primer in Diffusion of Innovations Theory*. Retrieved 1 March 2021 from http://www.rogerclarke.com/SOS/InnDiff.html

Cockburn, A. (2002). Agile Software Development Joins the" Would-Be" Crowd. *Cutter IT journal*, *15*(1), 6-12.

Cockburn, A., & Highsmith, J. (2001). Agile software development, the people factor. *Computer*, *34*(11), 131-133.

Cocosila, M., Archer, N., & Yuan, Y. (2009). Early investigation of new information technology acceptance: A perceived risk-motivation model. *Communications of the Association for Information Systems*, *25*(1), 30.

Cohen, D., Lindvall, M., & Costa, P. (2004). An introduction to agile methods. *Advances in Computers*, *62*, 1-66.

Cohn, M. (2006). *ScrumMaster: Appointed or Team-Selected?* https://www.mountaingoatsoftware.com/articles/scrummaster

Colantoni, A., Berardinelli, L., & Wimmer, M. (2020). DevopsML: Towards modeling devops processes and platforms. Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings,

Colavita, F. (2016). DevOps movement of enterprise agile breakdown silos, create collaboration, increase quality, and application speed. Proceedings of 4th International Conference in Software Engineering for Defence Applications: SEDA 2015,

Collier, J. (2020). *Applied structural equation modeling using AMOS: Basic to advanced techniques*. Routledge.

Collis, J., & Hussey, R. (2003). *Business research : a practical guide to undergraduate and postgraduate students* (2nd ed. ed.). Palgrave Macmillan Basingstoke.

Compeau, D., Higgins, C. A., & Huff, S. (1999). Social Cognitive Theory and Individual Reactions to Computing Technology: A Longitudinal Study. *MIS Quarterly*, *23*(2), 145-158. https://doi.org/10.2307/249749

Compeau, D. R., & Higgins, C. A. (1991). A social cognitive theory perspective on individual reactions to computing technology.

Cooper, H. (1988). Organizing knowledge syntheses: A taxonomy of literature reviews. *Knowledge in society*, *1*(1), 104-126.

Cooper, H. (2010). Research Synthesis and Meta-analysis: A Step-by-step Approach Sage Publications. *Thousand Oaks, CA*.

Creswell, J. W. (2009). *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches.* SAGE.

Creswell, J. W. (2012). *Qualitative inquiry and research design: Choosing among five approaches* (3rd ed.). Sage.

Creswell, J. W. (2013). *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage publications.

Creswell, J. W. (2014). Research design: Qualitative, quantitative, and mixed methods approaches 4th ed. In: Thousand Oaks, CA: Sage.

Creswell, J. W., & Clark, V. L. P. (2007). *Designing and conducting mixed methods research*. Wiley Online Library.

Creswell, J. W., Plano Clark, V. L., Gutmann, M. L., & Hanson, W. E. (2003). Advanced mixed methods research designs. *Handbook of mixed methods in social and behavioral research*, *209*, 240.

Crittenden, M. (2016). *The why's and how's of jelling teams*. https://critter.blog/2016/10/27/the-whys-and-hows-of-jelling-teams/#:~:text=Summary,it's%20hard%20not%20to%20succeed.

Cronbach, L. J. (1951). Coefficient alpha and the internal structure of tests. *Psychometrika*, *16*(3), 297-334. https://doi.org/10.1007/BF02310555

Cronin, P., Ryan, F., & Coughlan, M. (2008). Undertaking a literature review: a step-by-step approach. *British journal of nursing*, *17*(1), 38-43.

Cuellar, M. (2010). Assessing project success: Moving beyond the triple constraint.

Curran, P. J., West, S. G., & Finch, J. F. (1996). The robustness of test statistics to nonnormality and specification error in confirmatory factor analysis. *Psychological Methods*, *1*, 16-29.

Dago, D., Kablan, J., Justin, A., Alphonse, A., Hermann-Désiré, L., Dagnogo, D., . . . Malerba, G. (2021). Normality Assessment of Several Quantitative Data Transformation Procedures. *Biostat Biom Open Access J*, *10*(3), 53-67. https://doi.org/10.19080/BBOAJ.2021.10.555786

Davis, A. M. (1992). Operational prototyping: a new development approach. *IEEE Software*, *9*(5), 70-78. https://doi.org/10.1109/52.156899

Davis, F. D. (1985). *A technology acceptance model for empirically testing new end-user information systems: Theory and results* Massachusetts Institute of Technology, Sloan School of Management].

Davis, F. D. (1989). Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Quarterly*, 319-340.

Davis, F. D. (1993). User acceptance of information technology: system characteristics, user perceptions and behavioral impacts. *International journal of man-machine studies*, *38*(3), 475-487.

Davis, F. D., Bagozzi, R. P., & Warshaw, P. R. (1992). Extrinsic and intrinsic motivation to use computers in the workplace. *Journal of Applied Social Psychology*, *22*(14), 1111-1132. https://doi.org/10.1111/j.1559-1816.1992.tb00945.x

de França, B. B. N., Jeronimo, H., & Travassos, G. H. (2016). Characterizing DevOps by hearing multiple voices. Proceedings of the 30th Brazilian symposium on software engineering,

de Roos, W. (2020). Risk Management in a DevOps world. https://amazic.com/risk-management-in-a-devops-world/

de Winter, J., & Dodou, D. (2012). Factor recovery by principal axis factoring and maximum likelihood factor analysis as a function of factor pattern and sample size. *Journal of Applied Statistics*, *39*, 695-710. https://doi.org/10.1080/02664763.2011.610445

DeMarco, T., & Lister, T. (2001). Peopleware. In: 日経 BP 社.

Deng, N., Allison, J. J., Fang, H. J., Ash, A. S., & Ware, J. E. (2013). Using the bootstrap to establish statistical significance for relative validity comparisons among patient-reported outcome measures. *Health and quality of life outcomes*, *11*, 1-12.

Denney, A. S., & Tewksbury, R. (2013). How to write a literature review. *Journal of criminal justice education*, *24*(2), 218-234.

Desai, R., & Nisha, T. (2021). Best Practices for Ensuring Security in DevOps: A Case Study Approach. Journal of Physics: Conference Series,

Dhaduk, H. (2022). *Etsy DevOps Case Study: The Secret to 50 Plus Deploys a Day*. Retrieved 1 September 2022 from https://www.simform.com/blog/etsy-devops-case-study/

Diamantopoulos, A., & Siguaw, J. (2000). *Introducing LISREL* https://doi.org/10.4135/9781849209359

Dikert, K., Paasivaara, M., & Lassenius, C. (2016). Challenges and success factors for large-scale agile transformations: A systematic literature review. *Journal of Systems and Software*, *119*, 87-108.

Dillon, A., & Morris, M. (1996). User Acceptance of Information Technology: Theories and Models. *Annual Review of Information Science and Technology*, *31*.

Ding, L., Velicer, W. F., & Harlow, L. L. (1995). Effects of estimation methods, number of indicators per factor, and improper solutions on structural equation modeling fit indices. *Structural equation modeling: a multidisciplinary journal*, *2*(2), 119-143.

Dobra, A. (2021). *How Is Netflix SO GOOD at DevOps?* Retrieved 01 September 2022 from https://www.bunnyshell.com/blog/how-netflix-does-devops

Dodds, W. B., Monroe, K. B., & Grewal, D. (1991). Effects of price, brand, and store information on buyers' product evaluations. *Journal of Marketing Research*, *28*(3), 307-319.

Donca, I.-C., Stan, O. P., Misaros, M., Gota, D., & Miclea, L. (2022). Method for continuous integration and deployment using a pipeline generator for agile software projects. *Sensors*, *22*(12), 4637.

Drury, M., Conboy, K., & Power, K. (2012). Obstacles to decision making in Agile software development teams. *Journal of Systems and Software*, *85*(6), 1239-1254. https://doi.org/https://doi.org/10.1016/j.jss.2012.01.058

Dwivedula, R., & Bolloju, N. (2020). Transitioning from Plan-Driven Methods to Agile Methods-Preparation for a Systematic Literature Review. 2020 5th International Conference on Communication and Electronics Systems (ICCES),

Dybâ, T., & Dingsoyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and software technology*, *50*(9–10), 833-859. https://doi.org/http://dx.doi.org/10.1016/j.infsof.2008.01.006

Ebert, C., Gallardo, G., Hernantes, J., & Serrano, N. (2016). DevOps. *IEEE Software*, *33*(3), 94-100.

Ebnehoseini, Z., Tara, M., Tabesh, H., Dindar, F. H., & Hasibian, S. (2020). Understanding key factors affecting on hospital electronic health record (EHR) adoption. *J Family Med Prim Care*, *9*(8), 4348-4352. https://doi.org/10.4103/jfmpc.jfmpc_109_20

Englander, M. (2012). The interview: data collection in descriptive phenomenological human scientific research*. *Journal of Phenomenological Psychology*, *43*(1), 13-35.

Erich, F., Amrit, C., & Daneva, M. (2014). A mapping study on cooperation between information system development and operations. International Conference on Product-Focused Software Process Improvement,

Faber, F. (2020). Testing in DevOps. In *The Future of Software Quality Assurance* (pp. 27-38). Springer, Cham.

Fabrigar, L., Wegener, D., MacCallum, R., & Strahan, E. (1999). Evaluating the Use of Exploratory Factor Analysis in Psychological Research. *Psychological Methods*, *4*, 272. https://doi.org/10.1037/1082-989X.4.3.272

Fagan, M. H., Neill, S., & Wooldridge, B. R. (2008). Exploring the intention to use computers: An empirical investigation of the role of intrinsic motivation, extrinsic motivation, and perceived ease of use. *Journal of Computer Information Systems*, *48*(3), 31-37.

Faisal, S. M., & Idris, S. (2020). Innovation factors influencing the supply chain technology (SCT) adoption: diffusion of innovation theory. *International Journal of Social Science Research*, *2*(2), 128-145.

Farida, I., & Ardiansyah, W. (2022). Technology Acceptance Model Factors: Implications on Digital-Wallet on Interest to Buy in Franchise Business. *Golden Ratio of Marketing and Applied Psychology of Business*, *2*(2), 147-157.

Farroha, B. S., & Farroha, D. L. (2014, 6-8 Oct. 2014). A Framework for Managing Mission Needs, Compliance, and Trust in the DevOps Environment. 2014 IEEE Military Communications Conference,

Faustino, J., Adriano, D., Amaro, R., Pereira, R., & da Silva, M. M. (2022). DevOps benefits: A systematic literature review. *Software: Practice and Experience*.

Fichman, R. G. (1992). Information technology diffusion: A review of empirical research. ICIS,

Field, A. (2005). *Discovering statistics using SPSS, 2nd ed*. Sage Publications, Inc.

Field, A. (2013). *Discovering statistics using IBM SPSS statistics*. sage.

Fishbein, M., & Ajzen, I. (1975). *Belief, attitude, intention and behavior: An introduction to theory and research*.

Fitsilis, P., Tsoutsa, P., & Gerogiannis, V. (2018). Industry 4.0: Required personnel competences. *Industry 4.0*, *3*(3), 130-133.

Fitzgerald, B. (2012). Software Crisis 2.0. *Computer*, *45*(4), 89-91.

Fitzgerald, B., & Stol, K.-J. (2014). Continuous software engineering and beyond: trends and challenges. Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering,

Fitzgerald, B., & Stol, K.-J. (2015). Continuous software engineering: A roadmap and agenda. *Journal of Systems and Software*.

Fitzgerald, B., & Stol, K.-J. (2017). Continuous software engineering: A roadmap and agenda. *Journal of Systems and Software*, *123*, 176-189.

Flefil, A., Alawneh, L., & Albalas, F. (2022). DevOps Culture in Software Development Companies in Jordan. 2022 13th International Conference on Information and Communication Systems (ICICS),

Flora, D., Labrish, C., & Chalmers, R. (2012). Old and New Ideas for Data Screening and Assumption Testing for Exploratory and Confirmatory Factor Analysis. *Frontiers in psychology*, *3*, 55. https://doi.org/10.3389/fpsyg.2012.00055

Floyd, C. (1984). A Systematic Look at Prototyping.

Fornell, C., & Larcker, D. F. (1981). Structural equation models with unobservable variables and measurement error: Algebra and statistics. In: Sage Publications Sage CA: Los Angeles, CA.

Forsgren, N., Brown, A., Humble, J., Kersten, N., & Kim, G. (2017). State of DevOps Report. https://services.google.com/fh/files/misc/state-of-devops-2017.pdf

Fowler, M. (2005). *The New Methodology*. Retrieved 12/13/2014 from http://www.martinfowler.com/articles/newMethodology.html

Fowler, M., & Highsmith, J. (2001). The agile manifesto. *Software Development*, *9*(8), 28-35.

Fox, W., & Bayat, M. S. (2007). *A guide to managing research*. Juta.

Fusch, P. I., & Ness, L. R. (2015). Are we there yet? Data saturation in qualitative research. *The Qualitative Report*, *20*(9).

Gall, M., & Pigni, F. (2018). Leveraging DevOps for mission critical software. https://docplayer.net/148601937-Leveraging-devops-for-mission-critical-software.html

Gandomani, T. J., Zulzalil, H., abdul ghani, A. a., Md Sultan, A. B., & Sharif, K. (2014). How Human Aspects Impress Agile Software Development Transition and Adoption. *International Journal of Software Engineering and Its Applications*, *8*, 129-148. https://doi.org/10.14257/ijseia.2014.8.1.12

Gansser, O. A., & Reich, C. S. (2021). A new acceptance model for artificial intelligence with extensions to UTAUT2: An empirical study in three segments of application. *Technology in Society*, *65*, 101535.

Gao, S., Mokhtarian, P. L., & Johnston, R. A. (2008). Nonnormality of data in structural equation models. *Transportation Research Record*, *2082*(1), 116-124.

Garousi, V., Tarhan, A., Pfahl, D., Coşkunçay, A., & Demirörs, O. (2019). Correlation of critical success factors with success of software projects: an empirical investigation. *Software Quality Journal*, *27*(1), 429-493. https://doi.org/10.1007/s11219-018-9419-5

Gartner, I. (2015). *10 Things CIOs Need to Know About Agile Development*. Retrieved 20 March 2019 from https://www.gartner.com/en/newsroom/press-releases/2015-07-01-gartner-highlights-10-things-cios-need-to-know-about-agile-development

Gaskin, C. J., & Happell, B. (2014). On exploratory factor analysis: A review of recent evidence, an assessment of current practice, and recommendations for future use. *International journal of nursing studies*, *51*(3), 511-521. https://doi.org/https://doi.org/10.1016/j.ijnurstu.2013.10.005

Gene, K. (2014). *The Amazing DevOps Transformation of the HP Laserjet Firmware Team (Gary Gruver)*. Retrieved 01 September 2022 from https://itrevolution.com/the-amazing-devops-transformation-of-the-hp-laserjet-firmware-team-gary-gruver/

Giamattei, L., Guerriero, A., Pietrantuono, R., Russo, S., Malavolta, I., Islam, T., . . . Panojo, F. S. (2024). Monitoring tools for DevOps and microservices: A systematic grey literature review. *Journal of Systems and Software*, *208*, 111906. https://doi.org/https://doi.org/10.1016/j.jss.2023.111906

Gold, M. S., Bentler, P. M., & Kim, K. H. (2003). A Comparison of Maximum-Likelihood and Asymptotically Distribution-Free Methods of Treating Incomplete Nonnormal Data. *Structural equation modeling: a multidisciplinary journal*, *10*(1), 47-79. https://doi.org/10.1207/S15328007SEM1001_3

Gordon, V. S., & Bieman, J. M. (1995). Rapid prototyping: lessons learned. *IEEE Software*, *12*(1), 85-95. https://doi.org/10.1109/52.363162

Govil, N., Saurakhia, M., Agnihotri, P., Shukla, S., & Agarwal, S. (2020). Analyzing the behaviour of applying agile methodologies & DevOps culture in e-commerce web application. 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184),

Green, R., Mazzuchi, T., & Sarkani, S. (2010). Understanding the role of synchronous & asynchronous communication in agile software development and its effects on quality. *Journal of Information Technology Management*, *21*(2), 8-23.

Grinyer, A. (2007). Investigating adoption of agile software development methodologies in organisations. In *Agile Processes in Software Engineering and Extreme Programming* (pp. 163-164). Springer.

Groenewald, T. (2004). A phenomenological research design illustrated. *International journal of qualitative methods*, *3*(1), 42-55.

Guba, E. G. (1990). *The paradigm dialogue*. SAGE Publications.
Guba, E. G., & Lincoln, Y. S. (1994). Competing paradigms in qualitative research. *Handbook of qualitative research*, *2*(163-194), 105.
Gupta, V., Kapur, P. K., & Kumar, D. (2017). Modeling and measuring attributes influencing DevOps implementation in an enterprise using structural equation modeling. *Information and software technology*, *92*, 75-91.
Hair, J., Black, W., Babin, B., & Anderson, R. (2009). Multivariate Data Analysis 7th Edition Pearson Prentice Hall. In: JOUR.
Hamdani, M., & Butt, W. H. (2017). Success and failure factors in agile development. 2017 International Conference on Computational Science and Computational Intelligence (CSCI),
Hamilton, T. (2023). *What is Jenkins? Why Use Continuous Integration (CI) Tool?* https://www.guru99.com/jenkin-continuous-integration.html
Hardgrave, B., & Johnson, R. (2003). Toward an information systems development acceptance model: The case of object-oriented systems development. *Engineering Management, IEEE Transactions on*, *50*, 322-336. https://doi.org/10.1109/TEM.2003.817293
Hardgrave, B. C., Davis, F. D., & Riemenschneider, C. K. (2003). Investigating determinants of software developers' intentions to follow methodologies. *Journal of Management Information Systems*, *20*(1), 123-152.
Hardikar, S., Ahirwar, P., & Rajan, S. (2021). Containerization: cloud computing based inspiration Technology for Adoption through Docker and Kubernetes. 2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC),
Hart, C. (1998). *Doing a literature review: Releasing the social science research imagination*. Sage.
Hasan, B., & Ali, J. (2006). The impact of general and system-specific self-efficacy on computer training learning and reactions. *Journal of Management Information and Decision Sciences*, *9*(1), 17.
Hasselbring, W., Henning, S., Latte, B., Möbius, A., Richter, T., Schalk, S., & Wojcieszak, M. (2019, 25-26 March 2019). Industrial DevOps. 2019 IEEE International Conference on Software Architecture Companion (ICSA-C),
Hidayat, M. N. R., Latifah, L., & Tusyanah, T. (2023). Analyzing the factors affecting the use behavior of the SIRADI system through behavioral intention as the mediating variable using the UTAUT 2 model. *Sustainable Social Development*, *1*(2).
Hofstee, E. (2006). Constructing a good dissertation. *Johannesburg: EPE*.
Hohl, P., Klünder, J., van Bennekum, A., Lockard, R., Gifford, J., Münch, J., . . . Schneider, K. (2018). Back to the future: origins and directions of the "Agile Manifesto"–views of the originators. *Journal of Software Engineering Research and Development*, *6*(1), 1-27.
Hoogland, J., & Boomsma, A. (1998). Robustness Studies in Covariance Structure Modeling: An Overview and a Meta-Analysis. *Sociological Methods & Research*, *26*, 329-367. https://doi.org/10.1177/0049124198026003003
Hooper, D., Coughlan, J., & Mullen, M. (2008). Structural equation modelling: Guidelines for determining model fit. *Articles*, 2.
Hoque, S., De Brito, M. S., Willner, A., Keil, O., & Magedanz, T. (2017). Towards container orchestration in fog computing infrastructures. 2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC),
Hrusto, A., Engström, E., & Runeson, P. (2023). Towards optimization of anomaly detection in DevOps. *Information and software technology*, *160*, 107241. https://doi.org/https://doi.org/10.1016/j.infsof.2023.107241
Hsieh, H. F., & Shannon, S. E. (2005). Three approaches to qualitative content analysis. *Qual Health Res*, *15*(9), 1277-1288. https://doi.org/10.1177/1049732305276687
Hu, L. t., & Bentler, P. M. (1999). Cutoff criteria for fit indexes in covariance structure analysis: Conventional criteria versus new alternatives. *Structural equation modeling: a multidisciplinary journal*, *6*(1), 1-55.

Humble, J., & Farley, D. (2010). *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation (Adobe Reader)*. Pearson Education.

Hummel, M., Rosenkranz, C., & Holten, R. (2015). The role of social agile practices for direct and indirect communication in information systems development teams. *Communications of the Association for Information Systems*, *36*(1), 15.

Huq, M. S. (2017). *A quantitative inquiry into software developers' intentions to use agile scrum method* [Ph.D., Capella University]. ProQuest One Academic. United States -- Minnesota. https://ukzn.idm.oclc.org/login?url=https://www.proquest.com/dissertations-theses/quantitative-inquiry-into-software-developers/docview/1885098988/se-2?accountid=158225

Hurkens, J. (2019). Driving Feature Velocity through Cultural Change: Why DevOps puts Mindset before Technology. https://www.cloudops.com/blog/driving-feature-velocity-through-cultural-change-why-devops-puts-mindset-before-technology/

Hussaini, S. W. (2014). Strengthening harmonization of Development (Dev) and Operations (Ops) silos in IT environment through systems approach. 17th International IEEE Conference on Intelligent Transportation Systems (ITSC),

Hüttermann, M. (2012). *DevOps for developers*. Apress.

Imenda, S. (2014). Is there a conceptual difference between theoretical and conceptual frameworks? *Journal of social sciences*, *38*(2), 185-195.

Jacobson, I., Booch, G., & Rumbaugh, J. (1999a). The Unified Process. *IEEE Software*, *16*(3), 96-102. https://doi.org/https://doi.org/10.1109/MS.1999.10013

Jacobson, I., Booch, G., & Rumbaugh, J. (1999b). *The Unified Software Development Process*. Addison-Wesley. http://books.google.co.za/books?id=pdNAuAAACAAJ

Jain, R., & Vijayakumar Bharathi, S. (2021). Doctors' Perceptions on the Use of Internet of Things Medical Devices (IOT-MDs) for Anemic Pregnant Women: A TAM2 Study. *International Journal of Healthcare Information Systems and Informatics (IJHISI)*, *16*(1), 58-80. https://doi.org/10.4018/IJHISI.2021010104

Jameel, A. S., Karem, M. A., & Ahmad, A. R. (2022). Behavioral intention to use e-learning among academic staff during COVID-19 pandemic based on UTAUT model. Proceedings of International Conference on Emerging Technologies and Intelligent Systems: ICETIS 2021 (Volume 1),

Jha, A. V., Teri, R., Verma, S., Tarafder, S., Bhowmik, W., Kumar Mishra, S., . . . Philibert, N. (2023). From theory to practice: Understanding DevOps culture and mindset. *Cogent Engineering*, *10*(1), 2251758.

John, M. M., Olsson, H. H., & Bosch, J. (2021). Towards mlops: A framework and maturity model. 2021 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA),

Johnson, R. B., Onwuegbuzie, A. J., & Turner, L. A. (2007). Toward a definition of mixed methods research. *Journal of mixed methods research*, *1*(2), 112-133.

Jones, C. (1995). Patterns of large software systems: failure and success. *Computer*, *28*(3), 86-87. https://doi.org/10.1109/2.366170

Jonker, M. (2017). *DevOps Implementation Model for Large IT Service Organizations*

Joosten, D., Basten, D., & Mellis, W. (2011). *Measurement of Information System Project Success in Organizations - What Researchers Can Learn from Practice*.

Kadim, A., & Sunardi, N. (2023). Financial management system (QRIS) based on UTAUT model approach in jabodetabek. *International Journal of Artificial Intelligence Research*, *6*(1.1).

Kafle, N. (2013). Hermeneutic phenomenological research method simplified. *Bodhi: An Interdisciplinary Journal*, *5*. https://doi.org/10.3126/bodhi.v5i1.8053

Kamuto, M. B., & Langerman, J. J. (2017). Factors inhibiting the adoption of DevOps in large organisations: South African context. 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT),

Kapser, S., & Abdelrahman, M. (2020). Acceptance of autonomous delivery vehicles for last-mile delivery in Germany–Extending UTAUT2 with risk perceptions. *Transportation Research Part C: Emerging Technologies*, *111*, 210-225.

Karamitsos, I., Albarhami, S., & Apostolopoulos, C. (2020). Applying DevOps Practices of Continuous Automation for Machine Learning. *Information*, *11*, 363. https://doi.org/10.3390/info11070363

Karlstrom, D., & Runeson, P. (2005). Combining agile methods with stage-gate project management. *IEEE Software*, *22*(3), 43-49. https://doi.org/10.1109/MS.2005.59

Karunarathna, C., Jayasinghe, S., & Wijayanayaka, J. (2023, 29-29 June 2023). Determinants that drive the behavioural intention of employees in the IT industry to use CI/CD framework: A study based on Sri Lankan IT companies. 2023 International Research Conference on Smart Computing and Systems Engineering (SCSE),

Katal, A., Bajoria, V., & Dahiya, S. (2019). DevOps: Bridging the gap between Development and Operations. 2019 3rd International Conference on Computing Methodologies and Communication (ICCMC),

Kent, B., Beedle, M., Bennekum, A. v., Cockburn, A., Cunningham, W., Fowler, M., . . . Thomas, D. (2001). *Manifesto for Agile Software Development*. Retrieved 15th June from http://agilemanifesto.org/

Kess, P., & Haapasalo, H. (2002). Knowledge creation through a project review process in software production. *International Journal of Production Economics*, *80*(1), 49-55. https://doi.org/https://doi.org/10.1016/S0925-5273(02)00242-6

Khalifa, M., & Verner, J. (2000). Drivers for software development method usage. *Engineering Management, IEEE Transactions on*, *47*, 360-369. https://doi.org/10.1109/17.865904

Khan, A., & Shameem, M. (2020). Multi-criteria decision making taxonomy for DevOps challenging factors using analytical hierarchy process. *Journal of Software: Evolution and Process*, *32*. https://doi.org/10.1002/smr.2263

Khan, A. I., Qureshi, M., & Khan, U. A. (2012). A Comprehensive Study of Commonly Practiced Heavy & Light Weight Software Methodologies. *arXiv preprint arXiv:1202.2514*.

Khan, M. S., Khan, A. W., Khan, F., Khan, M. A., & Whangbo, T. K. (2022). Critical Challenges to Adopt DevOps Culture in Software Organizations: A Systematic Review. *IEEE Access*, *10*, 14339-14349.

Khoa, B. T., Ha, N. M., Nguyen, T. V. H., & Bich, N. H. (2020). Lecturers' adoption to use the online Learning Management System (LMS): Empirical evidence from TAM2 model for Vietnam. *HO CHI MINH CITY OPEN UNIVERSITY JOURNAL OF SCIENCE-ECONOMICS AND BUSINESS ADMINISTRATION*, *10*(1), 3-17.

Khoo, C. S., Na, J. C., & Jaidka, K. (2011). Analysis of the macro-level discourse structure of literature reviews. *Online Information Review*.

Kim, H.-Y. (2013). Statistical notes for clinical researchers: assessing normal distribution (2) using skewness and kurtosis. *Restorative dentistry & endodontics*, *38*(1), 52-54.

Kim, S. H. (2008). Moderating effects of Job Relevance and Experience on mobile wireless technology acceptance: Adoption of a smartphone by individuals. *Information & Management*, *45*(6), 387-393. https://doi.org/https://doi.org/10.1016/j.im.2008.05.002

Kimmond, R. (1995). Survey into the acceptance of prototyping in software development. Proceedings Sixth IEEE International Workshop on Rapid System Prototyping. Shortening the Path from Specification to Prototype,

Kipreos, M. (2019). *The Impact of Leadership Style on the Adoption of Agile Software Development: A Correlational Study* [D.B.A., Liberty University]. ProQuest One Academic. United States -- Virginia. https://ukzn.idm.oclc.org/login?url=https://www.proquest.com/dissertations-theses/impact-leadership-style-on-adoption-agile/docview/2313670904/se-2?accountid=158225

Kivunja, C. (2018). Distinguishing between Theory, Theoretical Framework, and Conceptual Framework: A Systematic Review of Lessons from the Field. *International Journal of Higher Education*, *7*, 44. https://doi.org/10.5430/ijhe.v7n6p44

Kline, R. B. (2005). Principles and practice of structural equation modeling, (2nd ed.). Guilford Press.

Kobayashi, O., Kawabata, M., Sakai, M., & Parkinson, E. (2006). Analysis of the interaction between practices for introducing XP effectively. Proceedings of the 28th international conference on Software engineering,

Koch, A. (2005). Agile Software Development: Evaluating the methods for your organisation. *Boston, MA: Artech House*.

Kraushaar, J. M., & Shirland, L. E. (1985). A Prototyping Method for Applications Development by End Users and Information Systems Specialists. *MIS Quarterly*, *9*(3), 189-197. https://doi.org/10.2307/248948

Kristinsson, R. (2015). Software Development with DevOps.

Kumar, S. (2015). Structure Equation Modeling Basic Assumptions and Concepts: A Novices Guide.

Kuppuswami, S., Vivekanandan, K., Ramaswamy, P., & Rodrigues, P. (2003). The effects of individual XP practices on software development effort. *ACM SIGSOFT Software Engineering Notes*, *28*(6), 6-6.

Kwateng, K. O., Atiemo, K. A. O., & Appiah, C. (2018). Acceptance and use of mobile banking: an application of UTAUT2. *Journal of Enterprise Information Management*.

Lakhani, A. (2023). All You Need to Know About Release Management in DevOps. https://www.peerbits.com/blog/complete-guide-to-release-management-in-devops.html

Lambert, T. (2011). *Cross sectional study of agile software development methods and project performance*. Nova Southeastern University.

Lancelot Miltgen, C., Popovič, A., & Oliveira, T. (2013). Determinants of end-user acceptance of biometrics: Integrating the "Big 3" of technology acceptance with privacy context. *Decision Support Systems*, *56*, 103-114. https://doi.org/https://doi.org/10.1016/j.dss.2013.05.010

Larman, C., & Basili, V. R. (2003). Iterative and incremental developments. a brief history. *Computer*, *36*(6), 47-56.

Le, T. M. H., Duong, T. N. M., Nguyen, T. S., Le, T. T. H., & Nguyen, T. T. H. (2022). Assessing Student's Adoption of E-Learning: An Integration of TAM and TPB Framework. *Journal of Information Technology Education: Research*, *21*, 297-335.

Leedy, P. D., & Ormrod, J. E. (2005). Practical research: Planning and design.

Leffingwell, D. (2007). *Scaling software agility: best practices for large enterprises*. Pearson Education.

Leffingwell, D. (2010). *Agile software requirements: lean requirements practices for teams, programs, and the enterprise*. Addison-Wesley Professional.

Lei, M., & Lomax, R. (2005). The Effect of Varying Degrees of Nonnormality in Structural Equation Modeling. *Structural equation modeling*, *12*, 1-27. https://doi.org/10.1207/s15328007sem1201_1

Leite, L., Rocha, C., Kon, F., Milojicic, D., & Meirelles, P. (2019). A survey of DevOps concepts and challenges. *ACM Computing Surveys (CSUR)*, *52*(6), 1-35.

Liker, J. K., & Sindi, A. A. (1997). User acceptance of expert systems: a test of the theory of reasoned action. *Journal of Engineering and Technology Management*, *14*(2), 147-173. https://doi.org/https://doi.org/10.1016/S0923-4748(97)00008-8

Limayem, M., Hirt, S. G., & Cheung, C. M. K. (2007). How Habit Limits the Predictive Power of Intention: The Case of Information Systems Continuance. *MIS Quarterly*, *31*(4), 705-737. https://doi.org/10.2307/25148817

Linberg, K. R. (1999). Software developer perceptions about software project failure: a case study. *Journal of Systems and Software*, *49*(2), 177-192. https://doi.org/https://doi.org/10.1016/S0164-1212(99)00094-1

Lincoln, Y. S., & Guba, E. G. (1985). *Naturalistic inquiry*. sage.

Lindvall, M., Basili, V., Boehm, B., Costa, P., Dangle, K., Shull, F., . . . Zelkowitz, M. (2002). Empirical findings in agile methods. *Extreme Programming and Agile Methods—XP/Agile Universe 2002*, 81-92.

Liu, Y., & Zhou, Y. (2017). *The Challenges and Mitigation Strategies of Using DevOps during Software Development*

Livermore, J. A. (2007). Factors that impact implementing an agile software development methodology. Proceedings 2007 IEEE SoutheastCon,

Lombardi, F., & Fanton, A. (2023). From DevOps to DevSecOps is not enough. CyberDevOps: an extreme shifting-left architecture to bring cybersecurity within software security lifecycle pipeline. *Software Quality Journal*, *31*(2), 619-654. https://doi.org/10.1007/s11219-023-09619-3

López-Peña, M. A., Díaz, J., Pérez, J. E., & Humanes, H. (2020). DevOps for IoT Systems: Fast and Continuous Monitoring Feedback of System Availability. *IEEE Internet of Things Journal*, *7*(10), 10695-10707. https://doi.org/10.1109/JIOT.2020.3012763

Lopez, K., & Willis, D. (2004). Descriptive Versus Interpretive Phenomenology: Their Contributions to Nursing Knowledge. *Qualitative health research*, *14*, 726-735. https://doi.org/10.1177/1049732304263638

Lopez, V., & Whitehead, D. (2013). Sampling data and data collection in qualitative research. *Nursing & midwifery research: Methods and appraisal for evidence-based practice*, *123*, 140.

Louho, R., & Kallioja, M. (2006). Factors affecting the use of hybrid media applications. *Graphic Arts in Finland*, *35*.

Lubis, M., Witjaksono, W., & Azizah, A. H. (2019). Implementation of Enterprise Resource Planning (ERP) using Integrated Model of Extended Technology Acceptance Model (TAM) 2: Case Study of PT. Toyota Astra Motor. 2019 7th International Conference on Cyber and IT Service Management (CITSM),

Luz, W. P., Pinto, G., & Bonifácio, R. (2019). Adopting DevOps in the real world: A theory, a model, and a case study. *Journal of Systems and Software*, *157*, 110384. https://doi.org/https://doi.org/10.1016/j.jss.2019.07.083

Lwakatare, L. E. (2017). DevOps adoption and implementation in software development practice: concept, practices, benefits and challenges.

Lwakatare, L. E., Kilamo, T., Karvonen, T., Sauvola, T., Heikkilä, V., Itkonen, J., . . . Lassenius, C. (2019). DevOps in practice: A multiple case study of five companies. *Information and software technology*, *114*, 217-230. https://doi.org/https://doi.org/10.1016/j.infsof.2019.06.010

Lwakatare, L. E., Kuvaja, P., & Oivo, M. (2016). An exploratory study of devops extending the dimensions of devops with practices. *ICSEA 2016*, *104*.

Maayan, G. D. (2023). *Vulnerability Management for DevOps Teams: A Practical Guide*. Retrieved 10 December 2023 from

MacCallum, R. C., Browne, M. W., & Sugawara, H. M. (1996). Power analysis and determination of sample size for covariance structure modeling. *Psychological Methods*, *1*(2), 130-149. https://doi.org/10.1037/1082-989X.1.2.130

Mahalakshmi, M., & Sundararajan, M. (2013). Traditional SDLC vs scrum methodology–a comparative study. *International Journal of Emerging Technology and Advanced Engineering*, *3*(6), 192-196.

Mamakou, X. J. (2023). Intentions to continue using agile methods: The case of the Greek banking sector. *Journal of Systems and Software*, *202*, 111685. https://doi.org/https://doi.org/10.1016/j.jss.2023.111685

Mannaro, K., Melis, M., & Marchesi, M. (2004). Empirical analysis on the satisfaction of it employees comparing xp practices with other software development methodologies. In *Extreme Programming and Agile Processes in Software Engineering* (pp. 166-174). Springer.

Mansor, Z., Yahya, S., & Arshad, N. (2011). *Success Determinants in Agile Software Development Methodology*.

Maree, K. (2007). *First steps in research* (1st ed ed.). Van Schaik Pretoria.
Maroukian, K., & Gulliver, S. (2020). Exploring the link between leadership and Devops practice and principle adoption. *Advanced Computing: An International Journal*, *11*(4).
Marsh, H. W., Hau, K.-T., & Wen, Z. (2004). In Search of Golden Rules: Comment on Hypothesis-Testing Approaches to Setting Cutoff Values for Fit Indexes and Dangers in Overgeneralizing Hu and Bentler's (1999) Findings. *Structural equation modeling: a multidisciplinary journal*, *11*(3), 320-341. https://doi.org/10.1207/s15328007sem1103_2
Marshall, C., & Rossman, G. B. (1989). Designing qualitative research. Newburv Park. In: CA: sage.
Martin, A., Biddle, R., & Noble, J. (2004). The XP customer role in practice: Three studies. Agile Development Conference, 2004,
Marto, A., Gonçalves, A., Melo, M., Bessa, M., & Silva, R. (2023). ARAM: A Technology Acceptance Model to Ascertain the Behavioural Intention to Use Augmented Reality. *Journal of Imaging*, *9*(3), 73. https://www.mdpi.com/2313-433X/9/3/73
Mascheroni, M. A., & Irrazábal, E. (2018). Continuous testing and solutions for testing problems in continuous delivery: A systematic literature review. *Computación y Sistemas*, *22*(3), 1009-1038.
Mason, M. (2010). Sample size and saturation in PhD studies using qualitative interviews. Forum qualitative Sozialforschung/Forum: qualitative social research,
Matyunina, M. V. (2019). Diffusion of Innovation in the Digital Economy. Selected Papers of the IV All-Russian scientific and practical conference with international participation" Information Systems and Technologies in Modeling and Control"(ISTMC'2019),
Maznorbalia, A. S., & Awalluddin, M. A. (2021). Users Acceptance of E-Government System in Sintok, Malaysia: Applying the UTAUT Model. *Policy & Governance Review*, *5*(1), 66-81.
McIntosh, C. N. (2007). Rethinking fit assessment in structural equation modelling: A commentary and elaboration on Barrett (2007). *Personality and Individual Differences*, *42*(5), 859-867. https://doi.org/https://doi.org/10.1016/j.paid.2006.09.020
McKenna, K. (2022). *Gradle vs. Maven: DevOps tools comparison*. Retrieved 20 August 2023 from https://www.techrepublic.com/article/gradle-vs-maven/
Merriam, S. B. (2002). *Qualitative research in practice : examples for discussion and analysis* (1st ed ed.). Jossey-Bass San Francisco.
Micceri, T. (1989). The unicorn, the normal curve, and other improbable creatures. *Psychological Bulletin*, *105*, 156-166. https://doi.org/10.1037/0033-2909.105.1.156
Millan, R. (2022). How User Interface Testing Can Fit into the CI/CD Pipeline. https://thenewstack.io/how-user-interface-testing-can-fit-into-the-ci-cd-pipeline/
Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, *63*(2), 81-97. https://doi.org/10.1037/h0043158
Miller, S., Siems, T., & Debroy, V. (2021). Kubernetes for cloud container orchestration versus containers as a service (CaaS): practical insights. 2021 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW),
Milly, N., Xun, S., Meena, M. E., & Cobbinah, B. B. (2021). Measuring mobile banking adoption in Uganda using the Technology Acceptance Model (TAM2) and perceived risk. *Open Journal of Business and Management*, *9*(01), 397.
Mishra, A., Abdalhamid, S., Mishra, D., & Ostrovska, S. (2021). Organizational issues in embracing Agile methods: an empirical assessment. *International Journal of System Assurance Engineering and Management*, *12*, 1420-1433.
Mishra, A., & Alzoubi, Y. I. (2023). Structured software development versus agile software development: a comparative analysis. *International Journal of System Assurance Engineering and Management*, *14*(4), 1504-1522.
Mishra, A., & Otaiwi, Z. (2020). DevOps and software quality: A systematic mapping. *Computer Science Review*, *38*, 100308. https://doi.org/https://doi.org/10.1016/j.cosrev.2020.100308

Mishra, D., Akman, I., & Mishra, A. (2014). Theory of Reasoned Action application for Green Information Technology acceptance. *Computers in human behavior*, *36*, 29-40. https://doi.org/https://doi.org/10.1016/j.chb.2014.03.030

Misra, S. C., Kumar, V., & Kumar, U. (2009). Identifying some important success factors in adopting agile software development practices. *Journal of Systems and Software*, *82*(11), 1869-1890. https://doi.org/http://dx.doi.org/10.1016/j.jss.2009.05.052

Misra, S. C., Kumar, V., & Kumar, U. (2010). Identifying some critical changes required in adopting agile practices in traditional software development projects. *International Journal of Quality & Reliability Management*, *27*(4), 451-474.

Mohammad, S. M. (2016). Continuous Integration and Automation. *International Journal of Creative Research Thoughts (IJCRT), ISSN*, 2320-2882.

Mohammad, S. M. (2018). Streamlining DevOps automation for Cloud applications. *International Journal of Creative Research Thoughts (IJCRT), ISSN*, 2320-2882.

Mohammad, S. M. (2019). DevOps Automation Advances IT Sectors with the Strategy of Release Management. *International Journal of Computer Trends and Technology (IJCTT)–Volume*, *67*.

Montoya, A. K., & Edwards, M. C. (2021). The Poor Fit of Model Fit for Selecting Number of Factors in Exploratory Factor Analysis for Scale Evaluation. *Educational and Psychological Measurement*, *81*(3), 413-440. https://doi.org/10.1177/0013164420942899

Moore, G. C., & Benbasat, I. (1991). Development of an Instrument to Measure the Perceptions of Adopting an Information Technology Innovation [Article]. *Information Systems Research*, *2*(3), 192-222. http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=4431054&site=ehost-live

Morgan, D. L. (2007). Paradigms lost and pragmatism regained methodological implications of combining qualitative and quantitative methods. *Journal of mixed methods research*, *1*(1), 48-76.

Mowad, A. M., Fawareh, H., & Hassan, M. A. (2022). Effect of Using Continuous Integration (CI) and Continuous Delivery (CD) Deployment in DevOps to reduce the Gap between Developer and Operation. 2022 International Arab Conference on Information Technology (ACIT),

Moyón, F., Soares, R., Pinto-Albuquerque, M., Mendez, D., & Beckers, K. (2020). Integration of security standards in devops pipelines: An industry case study. Product-Focused Software Process Improvement: 21st International Conference, PROFES 2020, Turin, Italy, November 25–27, 2020, Proceedings 21,

Mudadi, A., & Lotriet, H. H. (2023). An analysis of devops' impact on information technology organisations: a case study. *South African Journal of Industrial Engineering*, *34*(1), 155-167.

Mudarikwa, G., & Grace, T. D. (2018). *Agile system development methodologies usage and acceptance in South African banking firms: an exploratory analysis* Proceedings of the Annual Conference of the South African Institute of Computer Scientists and Information Technologists, Port Elizabeth, South Africa. https://doi.org/10.1145/3278681.3278711

Muhammad, A., Siddique, A., Quadri, N. N., Saleem, U., Abul Hasan, M., & Shahzad, B. (2021). Investigating Crucial Factors of Agile Software Development Through Composite Approach. *Intelligent Automation & Soft Computing*, *27*, 15-34. https://doi.org/10.32604/iasc.2021.014427

Mujumdar, A., Masiwal, G., & Chawan, P. (2012). Analysis of various Software Process Models. *International Journal of Engineering Research and Applications (IJERA)*, *2*, 2015-2021.

Mulaik, S., James, L., Alstine, J., Bennett, N., Lind, S., & Stilwell, C. (1989). Evaluation of Goodness-of-Fit Indices for Structural Equation Models. *Psychological Bulletin*, *105*, 430-445. https://doi.org/10.1037/0033-2909.105.3.430

Munassar, N. M. A., & Govardhan, A. (2010). A comparison between five models of software engineering. *International Journal of Computer Science Issues (IJCSI)*, *7*(5), 94.

Myers, M. (2004). *Qualitative Research in Information Systems*. Retrieved 12 March 2023 from https://misq.umn.edu/skin/frontend/default/misq/MISQD_isworld/index.html

Mykytyn, P. P., & Harrison, D. A. (1993). The Application of the Theory of Reasoned Action to Senior Management and Strategic Information Systems. *Information Resources Management Journal (IRMJ)*, *6*(2), 15-26. https://EconPapers.repec.org/RePEc:igg:rmj000:v:6:y:1993:i:2:p:15-26

Myrbakken, H., & Colomo-Palacios, R. (2017). *DevSecOps: A Multivocal Literature Review*. https://doi.org/10.1007/978-3-319-67383-7_2

Mysari, S., & Bejgam, V. (2020). Continuous integration and continuous deployment pipeline automation using Jenkins Ansible. 2020 International conference on emerging trends in information technology and engineering (IC-ETITE),

Nagpal, S., & Shadab, A. (2014). Literature review: Promises and challenges of devops. *University of Waterloo.[S22]*.

Narasimhulu, M., Mounika, D. V., Varshini, P., Amarendra, K., & Rao, T. R. K. (2023). Investigating the Impact of Containerization on the Deployment Process in DevOps. 2023 2nd International Conference on Edge Computing and Applications (ICECAA),

Nasehi, A. (2013). A Quantitative Study on Critical Success Factors in Agile Software Development Projects; Case Study IT Company.

Nasir, M. H. N., & Sahibuddin, S. (2011). Critical success factors for software projects: A comparative study. *Scientific Research and Essays*, *6*(10), 2174-2186.

Naumann, J. D., & Jenkins, A. M. (1982). Prototyping: The New Paradigm for Systems Development [Article]. *MIS Quarterly*, *6*(3), 29-44. http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=4679101&site=ehost-live

Naur, P. (1968). Software engineering-report on a conference sponsored by the NATO Science Committee Garimisch, Germany. *http://homepages. cs. ncl. ac. uk/brian. randell/NATO/nato1968. PDF*.

Naur, P., & Randell, B. (1969). Software Engineering: Report of a conference sponsored by the NATO Science Committee, Garmisch, Germany, 7-11 Oct. 1968, Brussels, Scientific Affairs Division, NATO.

Nayak, M. K., & Patra, M. R. (2001). Agile Project Management-Redefining the Role of Managers. *management*, *11*, 2.

Newman, S. (2015). *Building microservices*. " O'Reilly Media, Inc.".

Nguyen, D. S. (2016a). Success Factors That Influence Agile Software Development Project Success. *American Scientific Research Journal for Engineering, Technology, and Sciences*, *17*, 172-222.

Nguyen, D. S. (2016b). Workplace Factors that Shape Agile Software Development Team Project Success. *American Scientific Research Journal for Engineering, Technology, and Sciences*, *17*, 323-391.

Nielsen, P. A., Winkler, T. J., & Nørbjerg, J. (2017). Closing the IT development-operations gap: The DevOps knowledge sharing framework. Joint Proceedings of the BIR 2017 pre-BIR Forum, Workshops and Doctoral Consortium,

Niko, I. (2007). An Overview of Agile Software Development Methodology and Its Relevance to Software Engineering. *Jurnal Teknik Informatika dan Sistem Informasi*, *2*(1).

Nikolopoulos, F., & Likothanassis, S. (2018). A complete evaluation of the TAM3 model for cloud computing technology acceptance. OTM Confederated International Conferences" On the Move to Meaningful Internet Systems",

Nor, K., Abu-Shanab, E., & Pearson, J. (2008). Internet banking acceptance in Malaysia based on the theory of reasoned action. *Journal of Information Systems and Technology Management*, *5*. https://doi.org/10.4301/10.4301%2FS1807-17752008000100001

Norris, M., & Lecavalier, L. (2009). Evaluating the Use of Exploratory Factor Analysis in Developmental Disability Psychological Research. *Journal of autism and developmental disorders*, *40*, 8-20. https://doi.org/10.1007/s10803-009-0816-2

Null, C. (2022). *10 companies killing it at DevOps*. Retrieved 01 September 2022 from https://techbeacon.com/app-dev-testing/10-companies-killing-it-devops

Nunnally, J. N. Y., NY: McGraw-Hill; 1967. . (1967). Psychometric theory. *New York, NY: McGraw-Hill;*.

Ogala, J. O. (2022). A Complete Guide to DevOps Best Practices. *International Journal of Computer Science and Information Security (IJCSIS)*, *20*(2).

Oles, B. (2018). Cloud Database Features Comparison – Amazon RDS vs Google Cloud SQL. https://severalnines.com/blog/cloud-database-features-comparison-amazon-rds-vs-google-cloud-sql/

Oliveira, T., Thomas, M., Baptista, G., & Campos, F. (2016). Mobile payment: Understanding the determinants of customer adoption and intention to recommend the technology. *Computers in human behavior*, *61*, 404-414. https://doi.org/10.1016/j.chb.2016.03.030

Olszewska, M., & Waldén, M. (2015). DevOps meets formal modelling in high-criticality complex systems. Proceedings of the 1st international workshop on quality-aware DevOps,

Ortu, M., Murgia, A., Destefanis, G., Tourani, P., Tonelli, R., Marchesi, M., & Adams, B. (2016). The emotional side of software developers in JIRA. Proceedings of the 13th international conference on mining software repositories,

Osborne, J. (2015). What is Rotating in Exploratory Factor Analysis? *Assessment*, *20*, 1-8.

Osborne, J. W. (2002). Notes on the use of data transformations. *Practical Assessment, Research and Evaluation*, *8*, 6.

Österberg, G. (2020). A Systematic Literature Review on DevOps and its Definitions, Adoptions, Benefits and Challenges.

Otieno, O. C., Liyayla, S., & Odongo, B. C. (2015). Theoretical and Practical Implications of Applying Theory of Reasoned Action in an Information Systems Study. *Open Access Library Journal*, *Vol.02No.12*, 5, Article 68973. https://doi.org/10.4236/oalib.1102054

Overhage, S., & Schlauderer, S. (2012). How sustainable are agile methodologies? Acceptance factors and developer perceptions in scrum projects.

Özer, G., & Yilmaz, E. (2011). Comparison of the theory of reasoned action and the theory of planned behavior: An application on accountants' information technology usage. *African Journal of Business Management*, *5*(1), 50-58.

Ozkan, S., & Kanat, I. E. (2011). e-Government adoption model based on theory of planned behavior: Empirical validation. *Government Information Quarterly*, *28*(4), 503-513. https://doi.org/https://doi.org/10.1016/j.giq.2010.10.007

Ozmete, E., & Hira, T. (2011). Conceptual analysis of behavioral theories/models: Application to financial behavior. *European Journal of Social Sciences*, *18*, 386-404.

Palopak, Y., & Huang, S.-J. (2024). Perceived impact of agile principles: Insights from a survey-based study on agile software development project success. *Information and software technology*, *176*, 107552. https://doi.org/https://doi.org/10.1016/j.infsof.2024.107552

Palvia, P., & Nosek, J. (1992). An Empirical Evaluation of System Development Methodologies. *Information Resources Management Journal*, *3*. https://doi.org/10.4018/irmj.1990070103

Parashar, R. (2022). DevOps KPI Challenges – Continuous Monitoring Safeguards. *International Journal of Research in Engineering, Science and Management*, *5*(6), 200-203. https://journal.ijresm.com/index.php/ijresm/article/view/2196

Park, S., & Huh, J.-H. (2018). Effect of Cooperation on Manufacturing IT Project Development and Test Bed for Successful Industry 4.0 Project: Safety Management for Security. *Processes*, *6*, 88. https://doi.org/10.3390/pr6070088

Patel, U. A., & Jain, N. K. (2013). New idea in waterfall model for real time software development. *International Journal of Engineering Research & Technology (IJERT)*, *2*(4), 115.

Patton, M. Q. (2002). Qualitative research and evaluation methods. *London: Sage*.

Paule, C., Düllmann, T. F., & Van Hoorn, A. (2019). Vulnerabilities in continuous delivery pipelines? A case study. 2019 IEEE international conference on software architecture companion (ICSA-C),

Pavlou, P. (2002). What Drives Electronic Commerce? A Theory of Planned Behavior Perspective. *Academy of Management Proceedings*, *2002*. https://doi.org/10.5465/APBPP.2002.7517579

Pennington, J. (2019). The Eight Phases of a DevOps Pipeline. https://medium.com/taptuit/the-eight-phases-of-a-devops-pipeline-fda53ec9bba

Percival, G. (2020). *The Win(d)s of Change Management – DevSecOps*. Retrieved 15 August 2023 from https://itsm.tools/devsecops-winds-change-management/

Perera, P., Bandara, M., & Perera, I. (2016). Evaluating the impact of DevOps practice in Sri Lankan software development organizations. 2016 Sixteenth International Conference on Advances in ICT for Emerging Regions (ICTer),

Perera, P., Silva, R., & Perera, I. (2017). Improve software quality through practicing DevOps. 2017 Seventeenth International Conference on Advances in ICT for Emerging Regions (ICTer),

Perveez, S. H. (2023). What is Git: Features, Commands, Branch and Workflow in Git. https://www.simplilearn.com/tutorials/git-tutorial/what-is-git

Petersen, K., & Wohlin, C. (2009). A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case. *Journal of Systems and Software*, *82*(9), 1479-1490. https://doi.org/https://doi.org/10.1016/j.jss.2009.03.036

Peuraniemi, T. (2014). Review: DevOps, value-driven principles, methodologies and tools. Dataand Value-Driven Softw. Eng. with Deep Cust. *Insight*, *43*.

Pinto, A. S., Abreu, A., Costa, E., & Paiva, J. (2022). Augmented reality for a new reality: using UTAUT-3 to assess the adoption of mobile augmented reality in tourism (MART). *Journal of Information Systems Engineering and Management*, *7*(2), 14550.

Pinto, J., & Slevin, D. (1988). Project success: Definitions and measurement techniques. *Project Management Journal*, *2*.

PMI, A. (2008). Guide to the Project Management Body of Knowledge (PMBoK Guide 4th ed). *PMI, USA*.

Polit, D. F., & Beck, C. T. (2018). *Essentials of nursing research : appraising evidence for nursing practice* (Ninth edition ed.). Wolters Kluwer Philadelphia.

Pressman, R. S. (2010). *Software engineering: a practitioner's approach*. McGraw-Hill Higher Education. http://books.google.co.za/books?id=y4k_AQAAIAAJ

Priyadarsini, K., Raj, E. F. I., Begum, A. Y., & Shanmugasundaram, V. (2020). Comparing DevOps procedures from the context of a systems engineer. *Materials Today: Proceedings*.

Punjabi, R., & Bajaj, R. (2016). User stories to user reality: A DevOps approach for the cloud. 2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT),

Puppet_Dora. (2017). *State of DevOps Report 2017*. https://services.google.com/fh/files/misc/state-of-devops-2017.pdf

Purnama, I. W. J. W., & Ginardi, R. V. H. (2019). Analysis of Application Based on Cloud Computing in Banking Industries in Indonesia Using Technology Acceptance Model (TAM) 2 Method Case Study The National Private Banks in Surabaya and Bali Region. *IPTEK Journal of Proceedings Series*(5), 519-526.

Putra, R. D., & Samopa, F. (2018). Analysis of factors affecting the acceptance of Surabaya e-government service using technology acceptance model (TAM) 3: a case study of E-Lampid. Mathematics, Informatics, Science, and Education International Conference (MISEIC 2018),

Rajapakse, R. N., Zahedi, M., & Babar, M. A. (2021). An Empirical Analysis of Practitioners' Perspectives on Security Tool Integration into DevOps. Proceedings of the 15th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM),

Ramadan, N., & Rizk, N. (2015). Multi-Dimensional Success Factors of Agile Software Development Projects. *International Journal of Computer Applications (IJCA), ISSN 0975-8887, USA*, *118*, 23-30. https://doi.org/10.5120/20823-3453

Raman, A., Thannimalai, R., Rathakrishnan, M., & Ismail, S. N. (2022). Investigating the Influence of Intrinsic Motivation on Behavioral Intention and Actual Use of Technology in Moodle Platforms. *International Journal of Instruction*, *15*(1), 1003-1024.

Ramírez-Correa, P., Rondán-Cataluña, F. J., Arenas-Gaitán, J., & Martín-Velicia, F. (2019). Analysing the acceptation of online games in mobile devices: An application of UTAUT2. *Journal of Retailing and Consumer Services*, *50*, 85-93.

Ramler, R., Putschögl, W., & Winkler, D. (2014). Automated testing of industrial automation software: practical receipts and lessons learned. Proceedings of the 1st International Workshop on Modern Software Engineering Methods for Industrial Automation,

Ranatunga, R., HMS, P., & RGN, M. (2020). Methods and rules-of-thumb in the determination of minimum sample size when applying structural equation modelling: A review. *J Soc Sci Res*, *15*(2), 102-109.

Ravichandran, A., Taylor, K., & Waterhouse, P. (2016). *Devops for digital leaders: Reignite business with a modern devops-enabled software factory*. Springer Nature.

Reifer, D. J., Maurer, F., & Erdogmus, H. (2003). Scaling agile methods. *IEEE Software*, *20*(4), 12-14.

Reiners, G. (2012). Understanding the Differences between Husserl's (Descriptive) and Heidegger's (Interpretive) Phenomenological Research. *Journal of Nursing & Care*, *01*. https://doi.org/10.4172/2167-1168.1000119

Rennie, K. M. (1997). Exploratory and Confirmatory Rotation Strategies in Exploratory Factor Analysis.

Riemenschneider, C., Hardgrave, B., & Davis, F. (2002). Explaining software developer acceptance of methodologies: a comparison of five theoretical models. *Software Engineering, IEEE Transactions on*, *28*(12), 1135-1145.

Riungu-Kalliosaari, L., Mäkinen, S., Lwakatare, L. E., Tiihonen, J., & Männistö, T. (2016). DevOps Adoption Benefits and Challenges in Practice: A Case Study. Product-Focused Software Process Improvement: 17th International Conference, PROFES 2016, Trondheim, Norway, November 22-24, 2016, Proceedings 17,

Robson, C. (2002). Real World Research : A Resource for Social Scientists and Practitioner-Researchers / C. Robson.

Rockart, J. F. (1979). Chief executives define their own data needs. *Harvard business review*, *57*(2), 81-93. http://europepmc.org/abstract/MED/10297607

Rodrigues, G. d. S. (2016). *DevOps: the new development process*

Rodriguez, M., & Buyya, R. (2020). Container orchestration with cost-efficient autoscaling in cloud computing environments. In *Handbook of research on multimedia cyber security* (pp. 190-213). IGI global.

Rogers, E. M. (1962). *Diffusion of innovations*. Free Press of Glencoe.

Rogers, E. M. (1967). *Diffusion of Innovations*. New York: The Free Press of Glencoe.

Rogers, E. M. (1983). *Diffusion of innovations*. Free Press.

Romagny, P. (2024). *South Africa's digital banking is world class – here's why it leads*. Retrieved 10 January 2024 from https://www.bizcommunity.com/Article/196/513/245063.html

Rossman, G. B., & Wilson, B. L. (1985). Numbers and words: Combining quantitative and qualitative methods in a single large-scale evaluation study. *Evaluation review*, *9*(5), 627-643.

Rowse, M., & Cohen, J. (2021). A survey of DevOps in the south African software context. Proceedings of the 54th Hawaii International Conference on System Sciences,

Royce, W. W. (1970). Managing the development of large software systems. proceedings of IEEE, Los Angeles.

Ruparelia, N. B. (2010). Software development lifecycle models. *ACM SIGSOFT Software Engineering Notes*, *35*(3), 8-13.

Rütz, M. (2019). Devops: A systematic literature review. *no. August*, *23*, 25.

Ryan, R. M., & Deci, E. L. (2000). Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. *American Psychologist*, *55*(1), 68-78. https://doi.org/10.1037/0003-066X.55.1.68

Sachdeva, R. (2016). Automated testing in DevOps. Proc. Pacific Northwest Software Quality Conference,

Safeena, R., Date, H., Hundewale, N., & Kammani, D. A. (2013). Combination of TAM and TPB in Internet banking adoption. *International Journal of Computer Theory and Engineering*, *5*, 146-150. https://doi.org/10.7763/IJCTE.2013.V5.665

Sajannavar, S., & Dharwad, J. (2022). Conceptual Study of Agile Processes and Methodologies. *International Journal of Multidisciplinary Research Transactions*, *4*(9).

Salam, N. R. A., & Ali, S. (2020). Determining factors of cloud computing adoption: a study of Indonesian local government employees. *Journal of Accounting and Investment*, *21*(2), 312-333.

Salameh, H. (2019). The impact of devops automation, controls, and visibility practices on software continuous deployment and delivery. Proceedings of the 2nd International Conference on Research in Management and Economics,

Saldana, J. (2009). An introduction to codes and coding. *The coding manual for qualitative researchers*, 1-31.

Sambasivam, G. (2004). Extreme Programming (XP). *Agile Alliance, www. agilealliance. org/articles*.

Sánchez-Gordón, M., & Colomo-Palacios, R. (2018). Characterizing DevOps culture: a systematic literature review. Software Process Improvement and Capability Determination: 18th International Conference, SPICE 2018, Thessaloniki, Greece, October 9–10, 2018, Proceedings 18,

Saunders, M. N. (2011). *Research methods for business students, 5/e*. Pearson Education India.

Savor, T., Douglas, M., Gentili, M., Williams, L., Beck, K., & Stumm, M. (2016). Continuous deployment at Facebook and OANDA. Proceedings of the 38th International Conference on Software Engineering Companion,

Scaled_Agile. (2022). *DevOps*. Retrieved 2 September 2022 from https://alecoledelavie.com/accueil/vie_uploads/Portfolio_Programs_Projects_and%20BAU/PortFolio_stuff/Courses%20resources%20stuff/SAFe/SAFe%20A4/Appendix%20B%20-%20DevOps%20article-A4.pdf

Scaled_Agile. (2023). *SAFe 6 for Lean Enterprise* Retrieved 20 December 2023 from https://scaledagileframework.com/

Schach, S. R. (2008). *Object-oriented software engineering*. McGraw-Hill.

Schatz, B., & Abdelshafi, I. (2005). Primavera gets agile: a successful transition to agile development. *IEEE Software*, *22*(3), 36-42.

Schwaber, K. (1995). Scrum Development Process. OOPLSA'95 Workshop on Business. *Object Design and Implementation. Austin*.

Schwaber, K. (1997). Scrum development process. In *Business Object Design and Implementation* (pp. 117-134). Springer.

Schwaber, K. (2004). *Agile project management with Scrum*. Microsoft Press.

Schwaber, K. (2010). *Agile processes and self-organization*. http://www.controlchaos.com/my-articles/

Schwaber, K., & Beedle, M. (2002). *Agile software development with scrum. Series in agile software development* (Vol. 1). Prentice Hall Upper Saddle River.

Sekaran, U., & Bougie, R. (2010). *Research methods for business : a skill-building approach* (5th ed. ed.). Wiley ; Chichester : John Wiley [distributor].

Sen, A., Baumgartner, L., Heiß, K., & Wagner, C. (2021). DevOps paradigm-a pedagogical approach to manage and implement IT project. *Issues in Information Systems*, *22*(4).

Senapathi, M., Buchan, J., & Osman, H. (2018). DevOps capabilities, practices, and challenges: Insights from a case study. Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018,

Setiyani, L., Effendy, F., & Slamet, A. A. (2021). Using Technology Acceptance Model 3 (TAM 3) at Selected Private Technical High School: Google Drive Storage in E-Learning. *Utamax: Journal of Ultimate Research and Trends in Education*, *3*(2), 80-89.

Shah, J., & Dubaria, D. (2019). Building modern clouds: using docker, kubernetes & Google cloud platform. 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC),

Shah, J., Dubaria, D., & Widhalm, J. (2018). A Survey of DevOps tools for Networking. 2018 9th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON),

Shahin, M. (2015). Architecting for devops and continuous deployment. Proceedings of the ASWEC 2015 24th Australasian Software Engineering Conference,

Shahin, M., Ali Babar, M., & Zhu, L. (2017). Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices. *IEEE Access*, *PP*. https://doi.org/10.1109/ACCESS.2017.2685629

Shameem, M., Khan, A. A., Hasan, M. G., & Akbar, M. A. (2020). Analytic Hierarchy Process Based Prioritisation and Taxonomy of Success Factors for Scaling Agile Methods in Global Software Development. *IET Software*, *14*(4), 389-401. https://doi.org/https://doi.org/10.1049/iet-sen.2019.0196

Shane, J., Chan, T. J., & Mohan, Y. M. (2022). Factors affecting the intention to adopt e-wallet services during Covid-19 pandemic. *Journal of Arts & Social Sciences*, *5*(2), 28-40.

Sharma, S., & Coyne, B. (2015). DevOps for Dummies. 2nd IBM Limited Edition. In: John Wiley & Sons.

Sharma, S., Durand, R. M., & Gur-Arie, O. (1981). Identification and Analysis of Moderator Variables. *Journal of Marketing Research*, *18*(3), 291-300. https://doi.org/10.2307/3150970

Sharma, S., Sarkar, D., & Gupta, D. (2012). Agile Processes and Methodologies: A Conceptual Study. *International Journal on Computer Science and Engineering*, *4*.

Shenton, A. K. (2004). Strategies for ensuring trustworthiness in qualitative research projects. *Education for information*, *22*(2), 63-75.

Sheppard, B. H., Hartwick, J., & Warshaw, P. R. (1988). The Theory of Reasoned Action: A Meta-Analysis of Past Research with Recommendations for Modifications and Future Research. *Journal of Consumer Research*, *15*(3), 325-343. https://doi.org/10.1086/209170

Shimel, A. (2018). *DevOps Chat: The Modern Software Factory and CD Pipelines w/ Scott Willson, CA Automic*. https://devops.com/devops-chat-the-modern-software-factory-and-cd-pipelines-w-scott-wilson-ca-automic/

Shrestha, N. (2021). Factor analysis as a tool for survey analysis. *American Journal of Applied Mathematics and Statistics*, *9*(1), 4-11.

Siebra, C., Lacerda, R., Peixoto, J., Cerqueira, I., Da Silva, F., & Medeiros, A. (2018). From theory to practice: The challenges of a DevOps infrastructure as code implementation. ICSOFT 13th 2018, International Conference on Software Technologies. Porto: Portugal July,

Sommerville, I. (2007). *Software Engineering*. Addison-Wesley. http://books.google.co.za/books?id=B7idKfL0H64C

Soni, M. (2015, 25-27 Nov. 2015). End to End Automation on Cloud with Build Pipeline: The Case for DevOps in Insurance Industry, Continuous Integration, Continuous Testing, and Continuous Delivery. 2015 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM),

Sreyes, K., Davis, D., & Jayapandian, N. (2022). Internet of Things and Cloud Computing Involvement Microsoft Azure Platform. 2022 International Conference on Edge Computing and Applications (ICECAA),

Srivastava, A. (2023). Confluence vs. Jira. https://businessanalyst.techcanvass.com/confluence-vs-jira/#:~:text=Each%20is%20a%20star%20on,teams%20track%20and%20manage%20work.

Srivastava, S., Allam, K., & Mustyala, A. (2023). Software Automation Enhancement through the Implementation of DevOps.

Stahl, D., Martensson, T., & Bosch, J. (2017). Continuous practices and devops: beyond the buzz, what does it all mean? 2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA),

Standish, G. (2015). CHAOS report 2015. *The Standish Group International, Inc*, 1-13.
Standish, G. (2020). Standish Group Choas Report 2020. *The Standish Group International, Inc*. https://devfolio.co/projects/standish-group-chaos-report-pdf-download-7a9d
Stankovic, D., Nikolic, V., Djordjevic, M., & Cao, D.-B. (2013). A survey study of critical success factors in agile software projects in former Yugoslavia IT companies. *Journal of Systems and Software*, *86*, 1663–1678. https://doi.org/10.1016/j.jss.2013.02.027
Steiger, J. H. (2007). Understanding the limitations of global fit assessment in structural equation modeling. *Personality and Individual Differences*, *42*(5), 893-898. https://doi.org/https://doi.org/10.1016/j.paid.2006.09.017
Stettina, C. J., & Heijstek, W. (2011). Five agile factors: Helping self-management to self-reflect. European Conference on Software Process Improvement,
Stojanov, I., Turetken, O., & Trienekens, J. (2015). *A Maturity Model for Scaling Agile Development*. https://doi.org/10.1109/SEAA.2015.29
Straub, D., Boudreau, M.-C., & Gefen, D. (2004). Validation guidelines for IS positivist research. *Communications of the Association for Information Systems*, *13*(1), 24.
Straub, D. W. (1989). Validating Instruments in MIS Research. *MIS Quarterly*, *13*(2), 147-169. https://doi.org/10.2307/248922
Strode, D., Dingsøyr, T., & Lindsjorn, Y. (2022). A teamwork effectiveness model for agile software development. *Empirical Software Engineering*, *27*(2), 56.
Subramanya, R. A. (2016). *Devops vs Agile*. Retrieved 24 November 2024 from https://in.linkedin.com/in/ramesubra1963
Sudhakar, G. P. (2012). A model of critical success factors for software projects. *Journal of Enterprise Information Management*.
Sutherland, J. (2005). Future of scrum: Parallel pipelining of sprints in complex projects. Agile Development Conference (ADC'05),
Sutherland, J., & Schwaber, K. (2007). The scrum papers. *Nuts, Bolts and Origins of an Agile Process*.
Sutherland, J., Schwaber, K., Scrum, C.-c. O., & Sutherl, C. J. (2012). *The scrum papers: Nuts, bolts, and origins of an agile process*. Retrieved 11/11 from http://jeffsutherland.com/ScrumPapers.pdf
Swanson, R. A., & Chermack, T. J. (2013). *Theory building in applied disciplines*. Berrett-Koehler Publishers.
Tabachnick, B. G., & Fidell, L. S. (2007). Using multivariate statistics. 5. In: Boston: Allyn and Bacon.
Taherdoost, H. (2019). Importance of Technology Acceptance Assessment for Successful Implementation and Development of New Technologies. *Global Journal of Engineering Sciences*, *1*. https://doi.org/10.33552/GJES.2019.01.000511
Takeuchi, H., & Nonaka, I. (1986). The new new product development game. *Harvard business review*, *64*(1), 137-146.
Tamilmani, K., Rana, N. P., Prakasam, N., & Dwivedi, Y. K. (2019). The battle of Brain vs. Heart: A literature review and meta-analysis of "hedonic motivation" use in UTAUT2. *International Journal of Information Management*, *46*, 222-235. https://doi.org/https://doi.org/10.1016/j.ijinfomgt.2019.01.008
Tamilmani, K., Rana, N. P., Wamba, S. F., & Dwivedi, R. (2021). The extended Unified Theory of Acceptance and Use of Technology (UTAUT2): A systematic literature review and theory evaluation. *International Journal of Information Management*, *57*, 102269.
Tanner, M., & Seymour, L. (2014). The range and level of software development skills needed in the Western Cape, South Africa. Proceedings of the e-skills for knowledge production and innovation conference 2014, cape town, south africa,
Tashakkori, A., & Teddlie, C. (2003). SAGE Handbook of Mixed Methods in Social and Behavioral Research. In. https://doi.org/10.4135/9781506335193

Tate, G. (1990). Prototyping: helping to build the right software. *Information and software technology*, *32*(4), 237-244. https://doi.org/https://doi.org/10.1016/0950-5849(90)90056-W

Taylor, S., & Todd, P. A. (1995a). Assessing IT usage: The role of prior experience. *MIS Quarterly*, 561-570.

Taylor, S., & Todd, P. A. (1995b). Understanding information technology usage: A test of competing models. *Information Systems Research*, *6*(2), 144-176.

TechTarget. (2017). *An early adopter, Facebook is still leading DevOps best-practices*. Retrieved 01 September 2022 from https://www.techtarget.com/searchitoperations/opinion/An-early-adopter-Facebook-is-still-leading-DevOps-best-practices

Teixeira, D., Pereira, R., Henriques, T. A., Silva, M., & Faustino, J. (2020). A systematic literature review on DevOps capabilities and areas. *International Journal of Human Capital and Information Technology Professionals (IJHCITP)*, *11*(3), 1-22.

Templeton, G. F., & Byrd, T. A. (2003). Determinants of the Relative Advantage of a Structured SDM During the Adoption Stage of Implementation. *Information Technology and Management*, *4*(4), 409-428. https://doi.org/10.1023/A:1025186302598

Tessem, B. (2003, 2003//). Experiences in Learning XP Practices: A Qualitative Study. Extreme Programming and Agile Processes in Software Engineering, Berlin, Heidelberg.

Thammareddi, L., Kuppam, M., Patel, K., Marupaka, D., & Bhanushali, A. (2023). An extensive examination of the devops pipelines and insightful exploration. *International Journal of Computer Engineering and Technology*, *14*(3), 76-90.

Thompson, R. L., Higgins, C. A., & Howell, J. M. (1991). Personal Computing: Toward a Conceptual Model of Utilization. *MIS Quarterly*, *15*(1), 125-143. https://doi.org/10.2307/249443

Tipaldi, M., Götz, C., Ferraguto, M., Troiano, L., & Bruenjes, B. (2013). *The Robust Software Feedback Model: an Effective Waterfall Model Tailoring for Space SW*.

Tsai, Y. (2011). Relationship between organizational culture, leadership behavior and job satisfaction. *BMC health services research*, *11*(1), 1-9.

Tseng, H., Tu, P., Lee, Y., & Wang, T. (2013). A study of satellite navigation fleet management system usage in Taiwan with application of C-TAM-TPB model. *Information Technology Journal*, *12*(1), 15.

Tseng, T. H., Lin, S., Wang, Y.-S., & Liu, H.-X. (2022). Investigating teachers' adoption of MOOCs: the perspective of UTAUT2. *Interactive Learning Environments*, *30*(4), 635-650.

Tsoy, M., & Staples, D. S. (2020). Exploring critical success factors in Agile analytics projects. Proceedings of the 53rd Hawaii International Conference on System Sciences,

Vaasanthi, R., Kingston, S., & Kumari, V. (2017). Comparative Study of DevOps Build Automation Tools. *International Journal of Computer Applications*, *170*, 5-8.

Valaitis, R., Meagher-Stewart, D., Martin-Misener, R., Wong, S. T., MacDonald, M., & O'Mara, L. (2018). Organizational factors influencing successful primary care and public health collaboration. *BMC health services research*, *18*(1), 1-17.

Van Belzen, M., Trienekens, J., & Kusters, R. (2019). Critical success factors of continuous practices in a DevOps context.

Van Manen, M. (1984). " Doing" Phenomenological Research and Writing: An Introduction.

Vardhan, N. (2023). What is DevSecOps? https://www.opsmx.com/blog/what-is-devsecops/

Veiga, J. F., Floyd, S., & Dechant, K. (2001). Towards modelling the effects of national culture on IT implementation and acceptance. *Journal of Information Technology*, *16*(3), 145-158.

Venkatesh, V. (2000). Determinants of Perceived Ease of Use: Integrating Control, Intrinsic Motivation, and Emotion into the Technology Acceptance Model. *Information Systems Research*, *11*(4), 342-365. http://www.jstor.org/stable/23011042

Venkatesh, V., & Bala, H. (2008). Technology Acceptance Model 3 and a Research Agenda on Interventions. *Decision Sciences - DECISION SCI*, *39*, 273-315. https://doi.org/10.1111/j.1540-5915.2008.00192.x

Venkatesh, V., & Davis, F. D. (1996). A model of the antecedents of perceived ease of use: Development and test. *Decision Sciences*, *27*(3), 451-481.

Venkatesh, V., & Davis, F. D. (2000). A theoretical extension of the technology acceptance model: four longitudinal field studies. *Management Science*, *46*(2), 186-204.
Venkatesh, V., Morris, M., Davis, G., & Davis, F. (2003). User Acceptance of Information Technology: Toward a Unified View. *MIS Quarterly*, *27*(3), 425-478.
Venkatesh, V., & Speier, C. (1999). Computer technology training in the workplace: A longitudinal investigation of the effect of mood. *Organizational behavior and human decision processes*, *79*(1), 1-28.
Venkatesh, V., Thong, J., & Xu, X. (2016). Unified Theory of Acceptance and Use of Technology: A Synthesis and the Road Ahead. *Journal of the association for information systems*, *17*, 328–376. https://doi.org/10.17705/1jais.00428
Venkatesh, V., Thong, J. Y., & Xu, X. (2012). Consumer acceptance and use of information technology: extending the unified theory of acceptance and use of technology. *MIS Quarterly*, 157-178.
Verona, J. (2016). *Practical DevOps*. Packt Publishing Ltd.
Virmani, M. (2015a). *Understanding DevOps & Bridging the gap from Continuous Integration to Continuous Delivery* Fifth international conference on Innovative Computing Technology (INTECH 2015),
Virmani, M. (2015b). Understanding DevOps & bridging the gap from continuous integration to continuous delivery. Innovative Computing Technology (INTECH), 2015 Fifth International Conference on,
Vithana, N., Fernando, S., & Kapurubandara, M. (2015). Success Factors for Agile Software Development A Case Study from Sri Lanka. *International Journal of Computer Applications*, *113*, 10-18. https://doi.org/10.5120/19917-2056
Vuković, M., Pivac, S., & Kundid, D. (2019). Technology acceptance model for the internet banking acceptance in split. *Business Systems Research: International journal of the Society for Advancing Innovation and Research in Economy*, *10*(2), 124-140.
Wahaballa, A., Wahballa, O., Abdellatief, M., Xiong, H., & Qin, Z. (2015). Toward unified DevOps model. 2015 6th IEEE international conference on software engineering and service science (ICSESS),
Waldron, K. (2017). Implementing a Collaborative Working Environment Using Agile: the LexisNexis Experience. *Legal Information Management*, *17*, 16-19. https://doi.org/10.1017/S147266961700007X
Waller, J., Ehmke, N. C., & Hasselbring, W. (2015). Including performance benchmarks into continuous integration to enable DevOps. *ACM SIGSOFT Software Engineering Notes*, *40*(2), 1-4.
Wang, C. L., & Ahmed, P. K. (2004). The development and validation of the organisational innovativeness construct using confirmatory factor analysis. *European Journal of Innovation Management*, *7*(4), 303-313. https://doi.org/10.1108/14601060410565056
Wang, Y. S., Lin, H. H., & Luarn, P. (2006). Predicting consumer intention to use mobile service. *Information Systems Journal*, *16*(2), 157-179.
Waterman, A. (2004). *Diffusion of Innovations.* www.stanford.edu/class/symbsys205/Commentary-RogersDiffusionInnovations.html
Watts, S. (2020). *How & Why To Become a Software Factory*. https://www.bmc.com/blogs/software-factory/
Watts, S. (2023). DevOps Release Management Concepts & Best Practices. https://www.splunk.com/en_us/blog/learn/devops-release-management.html
Weber, I., Nepal, S., & Zhu, L. (2016). Developing Dependable and Secure Cloud Applications. *IEEE Internet Computing*, *20*(3), 74-79. https://doi.org/10.1109/MIC.2016.67
Webster, J., & Watson, R. T. (2002). Analyzing the Past to Prepare for the Future: Writing a Literature Review. *MIS Quarterly*, *26*(2), xiii-xxiii. http://www.jstor.org/stable/4132319

Wei, M.-F., Luh, Y.-H., Huang, Y.-H., & Chang, Y.-C. (2021). Young generation's mobile payment adoption behavior: Analysis based on an extended UTAUT model. *Journal of Theoretical and Applied Electronic Commerce Research*, *16*(4), 618-637.

West, S. G., Finch, J. F., & Curran, P. J. (1995). Structural equation models with non-normal variables: Problems and remedies.

Wettinger, J., Breitenbücher, U., & Leymann, F. (2014). Standards-based DevOps automation and integration using TOSCA. 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing,

Wheaton, B., Muthén, B., Alwin, D. F., & Summers, G. F. (1977). Assessing Reliability and Stability in Panel Models. *Sociological Methodology*, *8*, 84-136. https://doi.org/10.2307/270754

Wiedemann, A., Forsgren, N., Wiesche, M., Gewald, H., & Krcmar, H. (2019). Research for practice: the DevOps phenomenon. *Communications of the ACM*, *62*(8), 44-49.

Wiedemann, A., Wiesche, M., Gewald, H., & Krcmar, H. (2020). Understanding how DevOps aligns development and operations: a tripartite model of intra-IT alignment. *European Journal of Information Systems*, *29*(5), 458-473.

Wijaya, P. E., Rosyadi, I., & Taryana, A. (2019). An attempt to adopt DevOps on embedded system development: empirical evidence. Journal of Physics: Conference Series,

Williams, M. D., Rana, N. P., & Dwivedi, Y. K. (2015). The unified theory of acceptance and use of technology (UTAUT): a literature review. *Journal of Enterprise Information Management*.

Williams, M. L., Saunderson, I. P., & Dhoest, A. (2021). Students' Perceptions of the Adoption and Use of Social Media in Academic Libraries: A UTAUT Study. *Communicatio*, *47*(1), 76-94.

Wilson, V. (2016). Research methods: Content analysis.

Wisdom, J. P., Cavaleri, M. A., Onwuegbuzie, A. J., & Green, C. A. (2012). Methodological reporting in qualitative, quantitative, and mixed methods health services research articles. *Health services research*, *47*(2), 721-745.

Wood, W., Quinn, J. M., & Kashy, D. A. (2002). Habits in everyday life: thought, emotion, and action. *Journal of personality and social psychology*, *83*(6), 1281.

Woods, E. (2016). Operational: The Forgotten Architectural View. *IEEE Software*, *33*(3), 20-23. https://doi.org/10.1109/MS.2016.86

Wu, M.-Y., Chou, H.-P., Weng, Y.-C., & Huang, Y.-H. (2011). TAM-2 based study of website user behavior-using web 2.0 websites as an example. *WSEAS Transactions on Business and Economics*, *4*(8), 133-151.

Xuan, J., Duan, T., Guo, Q., Gao, F., Li, J., Qiu, X., & Wu, S. (2021). Microservice Publishing Technology Based on DevOps Architecture. 2021 IEEE 5th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC),

Yaghoobi, T. (2017). Prioritizing key success factors of software projects using fuzzy AHP. *Journal of Software: Evolution and Process*, *30*, e1891. https://doi.org/10.1002/smr.1891

Yang, H.-H., & Su, C.-H. (2017). Learner behaviour in a MOOC practice-oriented course: In empirical study integrating TAM and TPB. *International Review of Research in Open and Distributed Learning: IRRODL*, *18*(5), 35-63.

Yang, J., Shen, G., & Ho, M. (2009). An overview of previous studies in stakeholder management and its implications for construction industry. *Journal of Facilities Management*, *7*, 159-175. https://doi.org/10.1108/14725960910952532

Yarlagadda, R. T. (2018). Understanding DevOps & bridging the gap from continuous integration to continuous delivery. *Understanding DevOps & Bridging the Gap from Continuous Integration to Continuous Delivery', International Journal of Emerging Technologies and Innovative Research (www. jetir. org), ISSN*, 2349-5162.

Yarlagadda, R. T. (2019). How DevOps Enhances the Software Dévelopment Quality. *International Journal of Creative Research Thoughts (IJCRT), ISSN*, 2320-2882.

Yarlagadda, R. T. (2020). DevOps for Better Software Security in the Cloud. *DevOps for Better Software Security in the Cloud", International Journal of Emerging Technologies and Innovative Research (www. jetir. org), ISSN*, 2349-5162.

Yarlagadda, R. T. (2021). DevOps and its practices. *International Journal of Creative Research Thoughts (IJCRT), ISSN*, 2320-2882.

Yong, D. (2021). Design and practice of software architecture in agile development. Journal of Physics: Conference Series,

Yousafzai, S. Y., Foxall, G. R., & Pallister, J. G. (2010). Explaining Internet Banking Behavior: Theory of Reasoned Action, Theory of Planned Behavior, or Technology Acceptance Model? *Journal of Applied Social Psychology*, *40*(5), 1172-1202. https://doi.org/https://doi.org/10.1111/j.1559-1816.2010.00615.x

Yuen, K. F., Cai, L., Qi, G., & Wang, X. (2021). Factors influencing autonomous vehicle adoption: An application of the technology acceptance model and innovation diffusion theory. *Technology Analysis & Strategic Management*, *33*(5), 505-519.

Zacharis, G., & Nikolopoulou, K. (2022). Factors predicting University students' behavioral intention to use eLearning platforms in the post-pandemic normal: an UTAUT2 approach with 'Learning Value'. *Education and Information Technologies*, *27*(9), 12065-12082. https://doi.org/10.1007/s10639-022-11116-2

Zahid, H., Ali, S., Abu-Shanab, E., & Muhammad Usama Javed, H. (2022). Determinants of intention to use e-government services: An integrated marketing relation view. *Telematics and Informatics*, *68*, 101778. https://doi.org/https://doi.org/10.1016/j.tele.2022.101778

Zaib, S., & Lakshmisetty, P. K. (2021). A systematical literature review and industrial survey in addressing the possible impacts with the continuous testing and delivery during DevOps transformation.

Zainal, P., Razali, D., & Mansor, Z. (2020). Team formation for agile software development: a review. *Int. J. Adv. Sci. Eng. Inf. Technol*, *10*(2), 555-561.

Zaitsev, A., Gal, U., & Tan, B. (2018). Reviewing the role of the agile manifesto and agile methods in literature.

Zamfir, V.-A., Carabas, M., Carabas, C., & Tapus, N. (2019). Systems monitoring and big data analysis using the elasticsearch system. 2019 22nd International Conference on Control Systems and Computer Science (CSCS),

Zaydi, M., & Nassereddine, B. (2020). DevSecOps PRACTICES FOR AN AGILE AND SECURE IT SERVICE MANAGEMENT. *Journal of Management Information and Decision Sciences*, *23*(2), 1-16. https://ukzn.idm.oclc.org/login?url=https://www.proquest.com/scholarly-journals/devsecops-practices-agile-secure-service/docview/2424965737/se-2?accountid=158225

Zhang, X., Yu, P., Yan, J., & Spil, I. T. A. (2015). Using diffusion of innovation theory to understand the factors impacting patient acceptance and use of consumer e-health innovations: a case study in a primary care clinic. *BMC health services research*, *15*(1), 1-15.

Zhang, Y., & Wildemuth, B. M. (2005). Qualitative Analysis of Content by. *Human Brain Mapping*, *30*(7), 2197-2206.

Zhou, K., Taigang, L., & Lifeng, Z. (2015, 15-17 Aug. 2015). Industry 4.0: Towards future industrial opportunities and challenges. 2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD),

APPENDIX A – INTERVIEW SCHEDULE

General Instructions for the Interview

During the interview, you are at liberty to request clarification or repetition of the question. There is no time limit set for answering a particular question or for the duration of the interview session. It is advisable to complete the interview in a single sitting.

Demographic & Background Information:

Job Title/Position						
Type of Organisation						
Job Description						
Department						
Gender	***Male***			*Female*		
Qualification(s)	*Under_ graduate Degree/ Diploma*	*Post_ graduate Degree*	*Honours*	Masters	*PhD*	*Others- (please specify)*
Approximately how long have you been involved in software development?						
Approximately how long have you been involved in the use of Agile Software Development?						
Approximately how long have you been involved in the use of DevOps Software Development?						

Context for **PART A** of the Interview:

In this part of the interview, the questions are directed at your general perception of the **processes/methods used for DevOps** based on your experiential knowledge of DevOps (or of software development in general).

PART A – ***PHENOMENOLOGICAL KNOWLEDGE OF AGILE SOFTWARE DEVELOPMENT Methodology (ASDM) – A Bracketing Approach***	
1.	How would you define DevOps?
2.	Please provide some detail regarding your experience(s) of the processes/methods used with DevOps (or software development in general).
3.	What is your view of? • Software development in general • Agile software development – Scaled Agile Framework
4.	Does DevOps work best in conjunction with agile software development?

Context for **PART B** of the Interview (Knowledge & Practice):

In this part of the interview, the questions are directed at your perceptions of the generic activities involved in the DEVOPS SOFTWARE (DEVELOPMENT) PROCESS.

PART B – ***PHENOMENOLOGICAL KNOWLEDGE OF DEVOPS SOFTWARE DEVELOPMENT METHODOLOGY – A Hermeneutic Approach***	
1.	What do you think of the **DevOps process** from its inception to its current practice?
2.	Based on your **experience** of using the DevOps process, what are the strengths of this process?
3.	Based on your experience of using the DevOps, what are the weaknesses of this process?
4.	Based on your experience of using the DevOps, what are the challenges in adopting process?
5.	Prior to using DevOps what software development process did you follow?
6.	What were the problems in this process that lead to the adoption of the DevOps process?
7.	Was the DevOps process successful in overcoming the problems with the previous software process?

Context for **PART C** of the Interview:

In this part of the interview, the questions are directed at your perceptions of the success criteria for DevOps projects.

PART C – ***PHENOMENOLOGICAL KNOWLEDGE OF SUCCESS CRITERIA IN DEVOPS PROJECT – A Hermeneutic Approach***	
1.	How do you judge the success of a DevOps project?
2.	What are the success criteria for your company/team in a DevOps project?
3.	Do you think that different stakeholders have different perceptions of the DevOps success? If yes/no, please elaborate.
4.	How is the success criteria in DevOps software projects different to that from other type of projects…such as Agile and Waterfall projects?

Context for **PART D** of the Interview:

In this part of the interview, the questions are directed at your perceptions of the success factors for DevOps projects.

PART D – PHENOMENOLOGICAL KNOWLEDGE OF SUCCESS FACTORS IN DEVOPS PROJECT – A Hermeneutic Approach	
1.	Based your lived experienced with DevOps software development, how does of organisational factors (e.g. Strong executive support, committed sponsor/manager, face to face communication, etc.) affect DevOps project success?
2.	Based on your lived experienced with DevOps software development, how does people factors (e.g. Team members with high competence, teams members with great motivations, coherent, self-organising teamwork, etc) affect DevOps project success?
3.	Based on your lived experienced with DevOps software development, how does the DevOps process from a project management perspective (e.g. following DevOps oriented project management, following DevOps oriented configuration management process, strong communication with daily face to face meetings, etc.) affect DevOps project success?
4.	Based your lived experienced with DevOps software development, how does the technical perspective (e.g. well defined coding standards up front, pursuing simple design, right amount of documentation, etc.) affect DevOps project success?
5.	Based on your lived experienced with DevOps software development, how does the project factor from a risk oriented perspective (e.g. DevOps project non-life critical, project with dynamic accelerated schedule, projects with upfront risk analysis done, etc.) affect DevOps project success?
6.	What is your lived experienced with DevOps software development, how does automation (e.g. automated code review, automated testing, automated deployment, etc.) affect DevOps project success?

7.	Are there any others factors not mentioned above that could affect DevOps project success?
8.	Which factors that you feels are critical to the success of DevOps project?
9.	Which other factors that are important but may not be deemed as critical to the success of DevOps projects?

Context for **PART E** of the Interview:

In this part of the interview, the questions are directed at your perceptions of acceptance factors for DevOps projects.

PART E – PHENOMENOLOGICAL KNOWLEDGE OF ACCEPTANCE FACTORS IN DEVOPS PROJECT – A Hermeneutic Approach	
1.	Based on your lived experience, explain how the benefits (the degree to which the user expects that using DevOps will help him or her to attain gains in job performance) of using DevOps influences your intention to continue using DevOps.
2.	Based on your lived experience, explain how the ease associated with using DevOps influences your intention to continue using DevOps.
3.	Based on your lived experience, explain how other software professionals influences your intention to continue using DevOps.
4.	Based on your lived experience, explain how the availability of resources and support influences your intention to continue using DevOps.
5.	Based on your lived experience, explain how fun or pleasure derived from using DevOps influences your intention to continue using DevOps.
6.	Based on your lived experience, explain how using DevOps will become a habit that influences your intention to continue using DevOps.
7.	Are there any others factors not mentioned above that could affect DevOps acceptance?

Context for **PART F** of the Interview:

In this part of the interview, the questions are directed at your perceptions of DevOps in South Africa

PART F – AWARENESS OF DEVOPS IN SOUTH AFRICA, IT CHALLENGES AND FUTURE.	
1.	Do you think there is wide awareness of DevOps in South Africa?
2.	Do you think there is wide adoption of DevOps in the South African software development industry
3.	What factors you think are preventing wide spread adoption of DevOps in South Africa
4.	What impact will DevOps have on the fourth Industrial revolution
5.	What do you think of the future of DevOps?
	Anything additional you like to comment DevOps?

Thank You!

APPENDIX B – SURVEY INSTRUMENT

SURVEY INSTRUMENT SECTION A

DEMOGRAPHICAL INFORMATION:

*Please provide some **basic demographic information** by **ticking (✓)** the appropriate option:*

1. What is your age?

18-20	
21-25	
26-30	
31-35	
36-40	
41-45	
46-50	
Above 50	

2. What is your gender?

Male	
Female	
Rather not mention	

3. What is your highest level of tertiary education, which is ICT related?

None	
Certificate	
Diploma	
Degree	
Postgraduate Degree	
Other	

4. Type of organisation employed in

Software Development	
Banking	
Insurance	
Telecommunications	
Manufacturing	
Other (*Please specify*)	

*Please provide some basic information regarding your **involvement in DevOps Projects** by **ticking (✓)** the appropriate option:*

5. Which agile methodology do you use in your organisation or company?

Extreme Programming	
Scrum	
Dynamic Systems Development Method	
Feature-Driven Development	
Adaptive Software Development	
Lean software development	
Crystal	
Hybrid Approach (Please specify)	

6. What is the size of the project (number of project team members)?

5 or less team members	
6-10 team members	
11-15 team members	
More than 15 team members	

7. What is the length of the project (in months) which you normally do in your organisation?

Less than 3 months	
3 – 6 months	
7 - 9 months	
10 -12 months	
More than 12 months	

8. What is your job responsibility in the DevOps/Agile project?

Project manager	
Team lead	
System Analyst	
Developer	
Security	
DevOps Engineer	
Testing	
Operations	
Business Analyst	
Other (Please specify):	

9. What is your level of experience with DevOps software development projects (in years)?

1-2	
3-4	
5-6	
7-8	
Above 8	

10. How many DevOps software development project/features have you been involved with?

1-5	
6-10	
11-15	
16-20	
21-25	
Above 25	

11. How often do you use DevOps for software development?

Never	
Seldom	
Sometimes	
Often	
Almost Always	

SECTION B

SUCCESS AND ACCEPTANCE FACTORS OF THE DEVOPS PROJECT

(1 – Strongly disagree, 2 –Disagree, 3 – Neutral, 4 – Agree, 5 – Strongly agree)

ORGANISATIONAL FACTORS	**1**	**2**	**3**	**4**	**5**
1. The DevOps project team is collocated, i.e. all team members worked in the same location for ease of communication and casual, and constant contact.					
2. The organisation has a cooperative culture instead of hierarchal.					
3. The organisation has an oral culture placing high value on face-to-face communication style.					
4. DevOps methodology was universally accepted in the organisation.					
5. The organisation has a reward system that was appropriate for DevOps behaviour.					
6. The DevOps project team had a committed sponsor or a committed organisation manager.					
7. The DevOps project team received good management or executive support.					
PEOPLE DIMENSION	**1**	**2**	**3**	**4**	**5**
1. DevOps project management personnel has a ***positive relationship with the customer/business unit***					
2. DevOps project management personnel is ***knowledgeable in DevOps principles and processes***.					
3. The DevOps project team members work in a cohesive, self-organising teamwork manner and are committed to ensuring that the team achieves project success, i.e. relying on the collective ability of an autonomous team to solve problems and adapt to changing conditions.					
4. The DevOps project team members ***work in a cohesive, self-organising teamwork*** manner and are committed to the project success, i.e. relying on the collective ability of an autonomous team to solve problems and adapt to changing conditions.					

5. DevOps project management personnel adopts a "light-touch" or ***adaptive management style*** that encourages flexibility and creativity.					
6 The DevOps project team members have a ***high level of technical competence*** and expertise in areas such as problem solving, programming, security, testing and operations.					
7 DevOps team members are ***highly motivated***					
8. DevOps teams members are ***willing to learn and innovate***					
SECURITY FACTORS (1 – Strongly disagree, 2 – Disagree, 3 – Neutral, 4 – Agree, 5 – Strongly Agree)	**1**	**2**	**3**	**4**	**5**
1. DevOps projects use code analysis for vulnerability testing and quality assurance.					
2. DevOps projects use a change management process to ensure that any developer can suggest a mission critical security change					
3. DevOps projects compared new features and updates against evolving threats.					
4.DevOps projects incorporates security into the full development lifecycle thereby not sacrificing necessary security measures in the pursuit of CI/CD speed.					
5. DevOps projects conduct periodic scans and do code reviews and penetration tests even after code is released.					

PROCESS FACTORS (1 – Strongly disagree, 2 –Disagree, 3 – Neutral, 4 – Agree, 5 – Strongly agree)	**1**	**2**	**3**	**4**	**5**
1. The DevOps projects follow a ***planning and management process*** (that drive release roadmaps, project plans, and delivery schedules, as well as end-to-end traceability across these processes)					
2. The DevOps projects ***continuously integrated team members'*** *work* thereby reducing risk and identifying issues early in the software development life cycle.					
3. The DevOps projects follow the practice of ***Continuous delivery*** ensuring that the software can be released at any time					
4. The DevOps projects follow a practice of ***Continuous Testing*** (that consists of testing early, testing often and test everywhere) and ***automation*** thereby ***enabling continuous quality and improvement.***					
5. The DevOps projects were ***continuously monitored*** to for early warnings about operational and quality defects that may occur in production.					
PROJECT FACTORS (1 – Strongly disagree, 2 –Disagree, 3 – Neutral, 4 – Agree, 5 – Strongly agree)	**1**	**2**	**3**	**4**	**5**
1. The DevOps projects has a ***dynamic, accelerated schedule***.					
2. The DevOps projects has a ***variable scope*** that leads to emergent/new system requirements.					
3. The DevOps projects is characterised by ***detailed, upfront*** cost evaluations.					
4. The DevOps projects has ***up-front risk analysis*** done.					
5. The DevOps projects are characterised as ***business mission-critical software***.					

TOOLS and TECHNOLOGIES (1 – Strongly disagree, 2 –Disagree, 3 – Neutral, 4 – Agree, 5 – Strongly agree)	**1**	**2**	**3**	**4**	**5**
1. The DevOps projects use ***planning tools (such as Jira)*** for planning and project tracking allowing teams to ship code early and often.					
2. The DevOps projects use ***continuous integration and continuous delivery tools*** (such as Jenkins)					
3. The DevOps projects use ***Version Control tools*** (such as Git and GitHub) to coordinate work among programmers					
4. The DevOps projects use ***build tools*** (such as Gradle or Maven) to automate builds allowing for automatic download and configuration of dependencies or other libraries					
5. The DevOps projects use ***testing tools*** such as JUnit or Selenium which provides a framework for testing applications.					
6. The DevOps projects use ***operational tools*** *(such as Chef)* that treats infrastructure as code.					
7. The DevOps projects use ***deployment tools*** (such as Docker) which delivered software in packages called containers					
8. The DevOps projects used tools (such as Kubernetes) for ***container orchestration***.					
9. The DevOps projects use *monitoring tools* (such as Splunk) that provide real-time insight into errors across the infrastructure that helps troubleshoot and catch high- impact issues.					

ACTUAL SUCCESS OF THE DEVOPS SOFTWARE DEVELOPMENT PROJECTS **(1 – Strongly disagree, 2 –Disagree, 3 – Neutral, 4 – Agree, 5 – Strongly agree)**	**1**	**2**	**3**	**4**	**5**
1. DevOps projects are ***successful*** in terms of ***frequency of deployments***.					
2. DevOps projects are ***successful*** in terms of ***speed of deployments***.					
3. DevOps projects are ***successful*** in terms of ***speed of build verification***.					
4. DevOps projects are ***successful*** in terms of ***quality of the project outcome*** or of the ***resulting software product.***					
5. DevOps projects are ***successful*** in terms of ***deployment success rate***					
6. DevOps projects are ***successful*** in terms of ***meeting the customer/end user expectations*** of the system					

AUTOMATION FACTORS **(1 – Strongly disagree, 2 –Disagree, 3 – Neutral, 4 – Agree, 5 – Strongly agree)**	**1**	**2**	**3**	**4**	**5**
1. DevOps projects use ***automated execution*** for ***continuous testing***					
2. DevOps projects use an ***automated build trigger*** for ***continuous integration***					
3. DevOps projects use an ***automated environment configuration*** for ***integration, UI and regression testing***					
4. DevOps projects use ***automated performance*** and ***error notifications*** during ***continuous delivery***					

5. DevOps projects ***automatically*** provision for ***servers and installation of application code***					
CLOUD FACTORS **(1 – Strongly disagree, 2 –Disagree, 3 – Neutral, 4 – Agree, 5 – Strongly agree)**	**1**	**2**	**3**	**4**	**5**
1. DevOps projects use ***cloud platforms*** to provide ***advanced automation***					
2. DevOps projects use a ***centralised cloud platform*** for ***testing, deployment, monitoring and operating*** the production workloads					
3. DevOps projects use ***Infrastructure as Code*** for the ***provisioning of servers and installation of application code***.					
4. DevOps projects use provisioned ***multiple cloud based test environments*** to ***enhance quality and productivity*** during testing.					
CULTURE FACTORS **(1 – Strongly disagree, 2 –Disagree, 3 – Neutral, 4 – Agree, 5 – Strongly agree)**	**1**	**2**	**3**	**4**	**5**
1, DevOps teams have a ***culture of collaboration*** (where all team members are on the "same page" and work in support of the same outcome – the delivery of stable, high-quality software and systems)					
2. DevOps team have a culture that ***espouses a shared understanding*** between key role players (such as developers, quality assurance and operations) that ensure a ***sharing of responsibility*** for the software they build.					
3. DevOps team have a strong ***focus on continuous improvement*** to optimise performance, cost and speed of delivery.					
4. DevOps team has a ***culture of embracing failures*** so they can turn failures into opportunities to learn.					
5. DevOps teams have a ***culture to automate almost everything*** so that they can deploy software faster, safer and more reliably than ever.					

INTENTION FACTORS (**1 – Strongly disagree, 2 –Disagree, 3 – Neutral, 4 – Agree, 5 – Strongly agree**)	**1**	**2**	**3**	**4**	**5**
1. I ***intend to use a DevOps approach*** for software development projects on a regular basis.					
2. I ***intend to make an effort*** to practice a DevOps approach for software development projects on a regular basis.					
3. I ***predict that I will be able to use a DevOps approach*** for software development projects in my organisation					
4. Based on my ***knowledge of DevOps methodology***, I predict that I would make an effort to ***use it more often***					
5. The ***use of a DevOps approach*** will ensure the success of software projects					
PERFORMANCE EXPECTANCY FACTORS (**1 – Strongly disagree, 2 –Disagree, 3 – Neutral, 4 – Agree, 5 – Strongly agree**)	**1**	**2**	**3**	**4**	**5**
1. I find ***DevOps useful*** for software development and deployment.					
2. Using ***DevOps increases my chances of achieving things that are important*** during software development and deployment.					
3. ***Using DevOps*** helps me accomplish software development and deployment tasks ***quickly***.					
4. ***Using DevOps*** increases my ***productivity*** within a DevOps team.					
EFFORT EXPECTANCYFACTORS (**1 – Strongly disagree, 2 –Disagree, 3 – Neutral, 4 – Agree, 5 – Strongly agree**)	**1**	**2**	**3**	**4**	**5**
1. ***Learning*** how to use DevOps framework is ***easy for me***.					
2. My ***interaction*** with DevOps is ***clear and understandable.***					
3. I find DevOps ***easy to use*** when developing and deploying software.					

4. It is ***easy*** for me to ***become skillful*** in using DevOps.					
5. Using DevOps is ***easy and understandable*** in my company					
FACILITATING CONDITIONS (1 – Strongly disagree, 2 –Disagree, 3 – Neutral, 4 – Agree, 5 – Strongly agree)	**1**	**2**	**3**	**4**	**5**
I have the ***resources*** necessary to use DevOps strategy for software development.					
I have the ***knowledge*** necessary to use DevOps strategy for software development.					
The DevOps strategy is ***compatible*** with other software development processes that I use.					
HEDONIC MOTIVATION (1 – Strongly disagree, 2 –Disagree, 3 – Neutral, 4 – Agree, 5 – Strongly agree)	**1**	**2**	**3**	**4**	**5**
I am ***motivated to work*** since things get done faster and problem solved more efficiently and effectively.					
I ***enjoy learning*** from team members in other domains.					
Fun ***playing and investigating*** new tools to make the DevOps strategy better.					
I ***enjoy interaction*** with team members in other domains.					
HABIT (1 – Strongly disagree, 2 –Disagree, 3 – Neutral, 4 – Agree, 5 – Strongly agree)	**1**	**2**	**3**	**4**	**5**
It became a ***habit for me to collaborate*** with teams members from other IT domains					
It became a ***habit to look for ways to automate*** my software development processes					
It became a ***habit to help team members*** in other domains to fix things with a project when it fails					
It became a ***habit to research*** new tools and technologies					

Using the ***DevOps strategy*** has ***become natural*** to me					
SOCIAL INFLUENCE (1 – Strongly disagree, 2 –Disagree, 3 – Neutral, 4 – Agree, 5 – Strongly agree)	**1**	**2**	**3**	**4**	**5**
People who are ***important to me think that I should use DevOps*** for software development					
People who ***influence my behaviour think that I should use DevOps*** for software development.					
People whose **opinions that I value prefer that I use DevOps** for software development					

THANK YOU!!!

APPENDIX C – ETHICAL CLEARANCE

UNIVERSITY OF KWAZULU-NATAL
INYUVESI YAKWAZULU-NATALI

03 January 2024

Mr Mudaray Marimuthu (9261483)
School of Management, IT & Governance
Westville Campus

Dear M Marimuthu,

Protocol reference number: HSSREC/00000878/2019
Project title: A Critical Success Factors Framework for the Implementation of a DevOps Strategy for Software Development in the Banking Sector in South Africa
Amended title: A model for DevOps implementation in South Africa
Degree: PhD

Approval Notification – Amendment Application

This letter serves to notify you that your application and request for an amendment received on 19 December 2023 has now been approved as follows:

- Change in title

Any alterations to the approved research protocol i.e. Questionnaire/Interview Schedule, Informed Consent Form; Title of the Project, Location of the Study must be reviewed and approved through an amendment /modification prior to its implementation. In case you have further queries, please quote the above reference number.

PLEASE NOTE: Research data should be securely stored in the discipline/department for a period of 5 years.

HSSREC is registered with the South African National Health Research Ethics Council (REC-040414-040).

Best wishes for the successful completion of your research protocol.

Yours faithfully

[redacted]

.....
Professor Dipane Hlalele (Chair)

/ms

Humanities & Social Sciences Research Ethics Committee
UKZN Research Ethics Office Westville Campus, Govan Mbeki Building
Postal Address: Private Bag X54001, Durban 4000
Tel: +27 31 260 8350 / 4557 / 3587
Website: http://research.ukzn.ac.za/Research-Ethics/

APPENDIX D – QUANTITATIVE DATA

Table D.1: Wilcoxon Sign Test Hypothesis Test Summary

	Null Hypothesis	Test	Sig.[a,b]	Decision
1	The median of ORG equals 3,00.	One-Sample Wilcoxon Signed Rank Test	<,001	Reject the null hypothesis.
2	The median of SEC equals 3,00.	One-Sample Wilcoxon Signed Rank Test	<,001	Reject the null hypothesis.
3	The median of PEO equals 3,00.	One-Sample Wilcoxon Signed Rank Test	<,001	Reject the null hypothesis.
4	The median of PRO equals 3,00.	One-Sample Wilcoxon Signed Rank Test	<,001	Reject the null hypothesis.
5	The median of PROJ equals 3,00.	One-Sample Wilcoxon Signed Rank Test	<,001	Reject the null hypothesis.
6	The median of TECH equals 3,00.	One-Sample Wilcoxon Signed Rank Test	<,001	Reject the null hypothesis.
7	The median of AUT equals 3,00.	One-Sample Wilcoxon Signed Rank Test	<,001	Reject the null hypothesis.
8	The median of CLO equals 3,00.	One-Sample Wilcoxon Signed Rank Test	<,001	Reject the null hypothesis.
9	The median of CUL equals 3,00.	One-Sample Wilcoxon Signed Rank Test	<,001	Reject the null hypothesis.
10	The median of SUCC equals 3,00.	One-Sample Wilcoxon Signed Rank Test	<,001	Reject the null hypothesis.
11	The median of PE equals 3,00.	One-Sample Wilcoxon Signed Rank Test	<,001	Reject the null hypothesis.
12	The median of EE equals 3,00.	One-Sample Wilcoxon Signed Rank Test	<,001	Reject the null hypothesis.
13	The median of FC equals 3,00.	One-Sample Wilcoxon Signed Rank Test	<,001	Reject the null hypothesis.
14	The median of HM equals 3,00.	One-Sample Wilcoxon Signed Rank Test	<,001	Reject the null hypothesis.
15	The median of HAB equals 3,00.	One-Sample Wilcoxon Signed Rank Test	<,001	Reject the null hypothesis.
16	The median of SI equals 3,00.	One-Sample Wilcoxon Signed Rank Test	<,001	Reject the null hypothesis.
17	The median of INT equals 3,00.	One-Sample Wilcoxon Signed Rank Test	<,001	Reject the null hypothesis.

a. The significance level is ,050.
b. Asymptotic significance is displayed.

Model Summary and Parameter Estimates

Dependent Variable: INT

	Model Summary					Parameter Estimates	
Equation	R Square	F	df1	df2	Sig.	Constant	b1
Linear	,461	182,493	1	213	,000	1,731	,626

The independent variable is HAB.

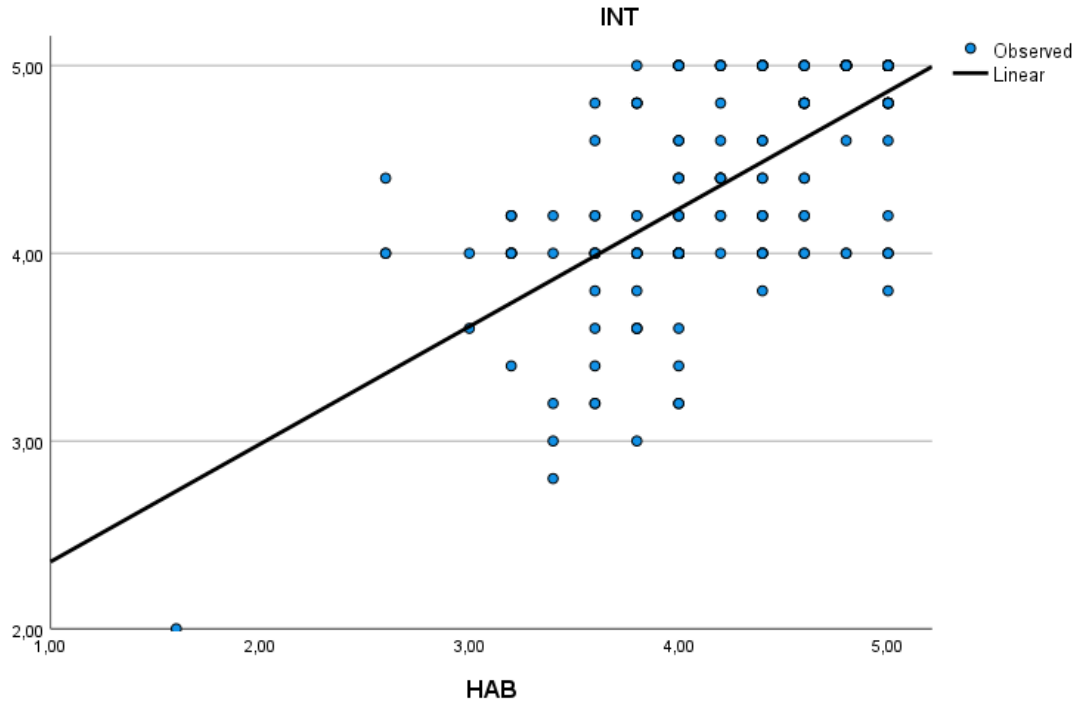

Figure D.1: Significant linearity test between Habit and Intention

Table D.2: Test for multicollinearity

Coefficients[a]								
Model		Unstandardized Coefficients		Standardized Coefficients	t	Sig.	Collinearity Statistics	
		B	Std. Error	Beta			Tolerance	VIF
1	(Constant)	,000	,283		-,001	,999		
	SEC	,129	,062	,128	2,074	,039	,567	1,763
	PEO	,108	,083	,083	1,297	,196	,524	1,908
	PRO	,089	,071	,080	1,250	,213	,524	1,908
	PROJ	,050	,063	,044	,793	,429	,712	1,404
	TECH	,270	,072	,252	3,721	,000	,468	2,136
	CLO	,206	,042	,260	4,838	,000	,746	1,341
	CUL	,164	,055	,177	3,006	,003	,622	1,608
	SI	,026	,045	,032	,586	,559	,721	1,387
	FC	-,011	,057	-,012	-,203	,840	,581	1,720
a. Dependent Variable: AUT								

Table D. 3: Covariance between error terms

Error terms	Modification index (M.I.) covariance
e83 <--> e84	42,784
e49 <--> e51	14,029
e46 <--> e47	18,027
e40 <--> e41	43,676
e32 <--> e33	29,295
e17 <--> e18	19,881
e10 <--> e11	28,248

Table D.4: Items loadings on many factors (snapshot of data)

			M.I.	Par Change
CUL1	<---	TECH1A	4,104	,125
CUL1	<---	TECH2	11,659	,140
CUL1	<---	AUT2	4,675	,094
CUL1	<---	AUT1	5,430	,096
CUL1	<---	SUCC5	4,354	,091
CUL1	<---	EFF1	5,535	,080
CUL1	<---	MOT4	4,344	,099
CUL1	<---	INT4	4,069	,098
MOT3	<---	AUT1A	5,539	,165
MOT3	<---	INTA	6,356	,162
MOT3	<---	AUT4	11,568	,146
MOT3	<---	SEC1	4,180	,089
MOT3	<---	TECH7	4,752	,098
MOT3	<---	CUL5	4,625	,096
MOT3	<---	AUT5	8,691	,123
MOT3	<---	SUCC5	4,781	,112
MOT3	<---	SUCC3	4,509	,096
MOT3	<---	PER2	6,359	,135
MOT3	<---	INT2	5,631	,134
MOT3	<---	INT1	9,411	,174
PROJ4	<---	TECH1	4,425	,114
PROJ4	<---	EFF4	4,367	-,090
PROJ4	<---	PER2	4,138	,117
PROJ4	<---	PER1	4,703	,132
PROJ3	<---	PROC	8,999	-,262
PROJ3	<---	ORG1A	4,970	-,165
PROJ3	<---	TECH1A	5,255	-,242
PROJ3	<---	SUCC1A	6,010	-,222
PROJ3	<---	PE1A	14,216	-,381
PROJ3	<---	MOT1A	5,245	-,275
PROJ3	<---	INTA	12,308	-,326
PROJ3	<---	PRO2	8,610	-,222
PROJ3	<---	PRO4	9,589	-,212
PROJ3	<---	ORG6	6,214	-,143
PROJ3	<---	SEC5	5,626	-,134
PROJ3	<---	SEC4	8,923	-,186
PROJ3	<---	TECH3	6,614	-,193

			M.I.	Par Change
PROJ3	<---	AUT2	5,072	-,168
PROJ3	<---	AUT1	8,666	-,206
PROJ3	<---	SUCC6	5,994	-,187
PROJ3	<---	SUCC4	8,233	-,203
PROJ3	<---	SUCC2	4,075	-,133
PROJ3	<---	PER4	12,514	-,282
PROJ3	<---	PER3	15,168	-,301
PROJ3	<---	PER1	13,714	-,303
PROJ3	<---	MOT4	4,324	-,168
PROJ3	<---	MOT2	4,048	-,168
PROJ3	<---	INT4	9,750	-,258
PROJ3	<---	INT3	8,341	-,232
PROJ3	<---	INT2	15,353	-,322
PROJ3	<---	INT1	8,033	-,233