COORDINATION OF DEVELOPMENT AND OPERATIONS ACTIVITIES IN AGILE SOFTWARE DEVELOPMENT

Ruth W. Macarthy

UNIVERSITY OF SALFORD

SCHOOL OF COMPUTING, SCIENCE AND ENGINEERING

JANUARY 26, 2023

This document is toward the degree of Doctor of Philosophy

Contents

List of Figures

List of Tables

Acronyms

CD Continuous Delivery

CI Continuous Integration

DSDM Dynamic System Development Method

IS Information Systems

ISD Information Systems Development

ISO International Organisation for Standardisation

IT Information Technology

OMG Object Management Group

Ops Operations

SC Situational Characteristics

SFIA Skills Framework for the Information Age

SME Situational Method Engineering

WIP Work in Progress

XP eXtreme Programming

Dedication

Memory is the treasury and guardian of all things - Marcus Tullius Cicero

This thesis is dedicated to the memories of my dear Father-in-law, Chief. (Engr) Osuonuokpenen Emmanuel Macarthy (JP), my sweet grandmother, Mrs. Obioma Fred Egbo, my aunt, Madam Mercy Fred Egbo (Ingilaye) and my gentle-spirited friend, Joyce Ishikaku Iwugo, who all passed on during my study. May their souls rest in peace.

Acknowledgement

Amen: Blessing, and glory, and wisdom, and thanksgiving, and honour, and power, and might, be unto our God for ever and ever. Amen – Revelation 7:12

My deepest thanks to my husband Motese Emmanuel Macarthy, for being a study mate and no.1 supporter; my loving parents, Pastor Dandy S. and Mrs Stella Mac'Odo, for their unwavering trust in me - supporting all my endeavours (no matter how daring or small); my siblings: Seiya, Tonye, Daniel, Tutumola, Tari, and Owebi for the many ways you cheered me on through this study.

I thank the Petroleum Technology Development Fund (PTDF) for the opportunity to carry out this research and the immense support through the journey.

My profound gratitude to my supervisor Prof. Julian Bass for providing inspiration and guidance, challenging me to be my best, showing unlimited patience, trusting my ideas, and supporting me through the trying times of the study. I wholeheartedly thank Dr. Tarek Gaber, my second supervisor, for his unconditional support.

My sincere thanks to all those who supported me through this process; friends, colleagues, peer reviewers, IA and IE assessors, examiners, practitioners who contributed to this study. I am most grateful.

DECLARATION OF ORIGINALITY

I hereby declare that work presented in this thesis is original and the outcome of a PhD research carried out by me.

I certify that, to the best of my knowledge, this thesis is consistent with copyright requirements. All views, techniques, and quotations used from other studies have been fully acknowledged with references. This thesis has not been previously submitted for the award of any academic degree at the University of Salford or any other university.

I further declare that I have dedicated full time to carry out this research under the guidance of my supervisor and in line with the regulation of the University of Salford.

Ruth Wakeni Macarthy

ABSTRACT

DevOps is described as a software engineering culture and philosophy that utilises cross-functional teams, to build test and release software faster and more reliably through automation. As an emerging concept, its definitions and best practices are still ambiguous. However, interest in DevOps and its adoption continue to rise significantly among industry practitioners. The unclear nature of the concept presents organisations with a wide range of unstructured choices, and few guidelines to navigate through a plethora of valuable information. To contribute to understanding of the subject and to a more structured implementation, this study employs a mixed method qualitative approach to investigates the practice of DevOps in four phases.

Phase one investigates the perceptions of DevOps and its associated practices, based on interviews with 11 industry practitioners across nine organisations. Phase 2 critically examines the DevOps implementation through interviews with 14 practitioners who lead DevOps transformation in their respective organisations. In both phases, transcripts of interviews were coded and analysed using a method informed by grounded theory.

The first phase identifies four different modes of DevOps implementation. A novel taxonomy is presented, which maps the approaches to cloud and on-premises deployment environments. In phase 2, six strategies of DevOps implementation were identified and uniquely characterised, with a critical examination of the roles of skillset and automation in the strategies. A combination of literature and theories generated from our data analysis led to the extension of an existing situational method engineering model, to create a novel model for DevOps implementation in phase three.

The model is evaluated by engaging with expert practitioners in a Focus Group workshop. The evaluation shows that the model provides clarity and better understanding DevOps implementation to practitioners. Arising from the workshop, a physical instantiation of the model was created in a repository. This version-controlled repository provides practitioners with the opportunity to collaboratively determine their DevOps strategy and keep track of the improvement journey.

This thesis concludes that organisations can implement DevOps in a structured and well-informed manner following the guidelines provided by our model.

Chapter 1

Introduction

1.1 Problem Context

The software industry is driven by the need for faster deployment of quality products. As software development is a complex socio-technical process, its delivery is dependent on the collaborative effort of people and teams with various skill sets and levels of expertise (Sharp & Robinson, 2008; Dingsøyr et al., 2012). Sydow et al. (2004) described the integration of such teams, to work as a unified whole, as coordination. Inter-dependencies that require synchronisation, feedback, information-sharing, and conflict resolution demand optimal management of activities in software development. According to Rodrigues et al. (2020); Van de Ven et al. (1976) effective coordination drives productivity. The pursuit of this goal of 'coordination' has necessitated the evolution of software development methodologies from the waterfall model to various forms of agile methods in recent times, along with associated practices. This continuous evolution is aimed at optimal service delivery. Organisations in the software development industry strive to keep up to date with these changes to maintain or increase market shares, optimise profit, and meet ever-growing customers' expectations. One of such emerging concepts is DevOps (Development and Operations).

DevOps is described as “a set of practices that advocate the collaboration between software developers and IT operations where the aim is to shorten the feedback loop while aligning the goals of both the development and IT operations departments” (Hüttermann, 2012). The concept is centred around aligning the goals of previously siloed teams and extending agile practices to IT operations activities. The strategy promises benefits such as faster delivery, improved quality and security, and better collaboration. Global giants like Spotify, Amazon, Netflix, LinkedIn, and Google have adopted DevOps and reported improved software delivery (Erich et al., 2017; Díaz et al., 2018; Wiedemann et al., 2019). Transitioning to DevOps may not be an easy task for organisations (Gill et al., 2018a). Firstly, existing literature claim that information around the concept and its adoption is still quite unstructured and scattered (Leite et al., 2019). This is to be expected as it is a developing movement and there is no accepted definition of DevOps. The scarcity of research around guidelines for its implementation poses a challenge for organisations seeking to adopt DevOps, as they must weave through an overwhelmingly diverse body of information and string together DevOps practices for themselves. Although work has been done around DevOps adoption, an in-depth investigation into its implementation in practice is still lacking in literature (Leite et al., 2019).

Tailoring of practices is quite common and encouraged in software development. Even with well-established methodologies, studies report differing outcomes of similar tailored methodologies. Ståhl & Bosch (2014) showed how the variant implementation and interpretation of continuous integration (CI) in practice can cause a disparity in the realisation of its adoption goals. This perceived variability agrees with the study by Bass & Haxby (2019), which identified three groups of activities that must be managed by the product owners in a large-scale software development (SD) project: scale, distance, and governance. Tailoring DevOps implementation is even riskier as it is an emerging concept with no consensus on its definition and its practices.

1.2 Research Motivation

Effective coordination of activities and the interrelationship among the actors is key in software development. In practice, coordination is tailored to suit organisations' specific needs. As consequence, outcomes from the adoption of similar methodologies are mostly unpredictable. According to a DevOps survey, although there is an increasing interest in research into DevOps, subjects such as the pathway to implementation and organisational factors influencing such decisions are still less studied than investigations into its nature and value derived from its adoption (Leite et al., 2019). In their work, Leite et al. (2019) stated that while engineers need to learn how to re-engineer their systems and qualify themselves for DevOps-related positions, managers seek to know how to introduce DevOps to their organisations and how to assess the quality of already adopted DevOps practice. Luz et al. (2019) claim that "It is still unclear how one could leverage such rich, yet scattered, information in an organised and structured way to properly adopt DevOps." This stance agrees with Gill et al. (2018b) insistence on "a need for clear understanding and guidelines to support effective DevOps adoption for ISs"

The forgoing motivates this research to identify existing patterns of work coordination in software development, investigate the factors driving such patterns, and develop a model to guide DevOps implementation. I believe this would contribute to the understanding of tailoring DevOps and improve the chances for its successful implementation.

1.3 Aim, Objectives and Research Questions

1.3.1 Aim

This research aims to investigate the patterns of work coordination between development and operations activities in organisations and propose a model for a structured DevOps implementation. For this study, the term “development” is used to describe the activities required to create an application from designing, programming, documenting, and testing, to bug fixing to meet users’ needs. Operations describe activities involving the deployment, performance monitoring, and management of software applications.

1.3.2 Objectives

This study attempts to investigate work coordination between development and operations teams, and factors that influence the approaches taken by organisation by:

1. Identifying approaches and strategies used to achieve work coordination in DevOps
2. Investigating the elements that influence the approach to DevOps implementation taken by organisations in the study.
3. Developing and evaluating an adaptive model to guide successful DevOps implementation, based on the identified relationships between the “elements of influence”, and emerging theories.

1.3.3 Research Questions

The chosen research methodology (Grounded Theory) does not encourage having preconceived research problems which can cause the research to be influenced by existing literature (Hoda et al., 2010), but rather allows the emergence of concepts (Glaser, 1978, 1998). This study chose an investigation into the rationale and evidence-based decision-making in organisational DevOps implementation, and the elements of influence as a general area of interest. Two main questions emerged from the investigation for which the research provides answers. The sub-questions provide finer details to the inquiry. The research aims to answer the questions:

Research Question (RQ)1: "How do practitioners describe and implement DevOps in practice?"

- **RQ1a**: What are practitioners' perceptions of DevOps definition and description?
- **RQ1b**: How are DevOps functions different from IT Operations and development teams' functions?
- **RQ1c**: How is DevOps implemented in practice?
- **RQ1d**: What strategies are employed in DevOps implementation?

Research Question 2: "How can organisations tailor DevOps implementation to suit organisational context?"

- **RQ2a**: What elements influence the approach taken by organisations to implement DevOps?
- **RQ2b**: How can DevOps implementation be tailored to suit on organisational context?

The study is divided into four (4) phases to address the research questions.

Phase 1: Data collection and analysis to understand practitioners' perception of DevOps (RQ1a, RQ1b, RQ1c).

Phase 2: Data collection and analysis of DevOps implementation to examine the strategies employed and the determinants of the strategies (RQ1d, RQ2a).

Phase 3: Development of model based on findings (of phases 1 and 2) and comparison with literature (RQ2b).

Phase 4: Model Evaluation.

1.4 Research Contribution

This research is an in-depth study of the approaches to work coordination in agile software development and the factors of influence. Specifically, the contributions of the study to knowledge and practice are as follows:

1. At the end of phase 1, I provided an empirical taxonomy of DevOps implementation. This classification describes developers' interaction with On-premises IT operations, Outsourced Ops, DevOps teams, and DevOps bridge teams. The taxonomy is a novel mapping of the identified approaches to on-premises and cloud-based deployments, and the facilitators of DevOps practices in the different approaches. Furthermore, I identified three distinct groups of activities in the fourth mode: provisioning and maintenance of physical systems, function virtualization and creation of automated pipelines, and development, deployment, and maintenance of applications, which (I believe) may have given rise to the implementation of DevOps as bridge teams.
2. A further investigation into the approaches to DevOps taken by the organisations in the study led to the identification of 6 strategies used by organisations to implement DevOps: Platform, Greenfield Application, Monolith Decomposition, Process Improvement, Cultural Improvement, and Advocacy. Except for the platform strategy which has previously been explored in literature, no study was found that identified or investigated the other 5 strategies. Following the guidelines of Grounded Theory, phase 2 culminated in the theory that a striking correlation exists between the skillset and the strategy adopted by organisations to implement DevOps.
3. The generation of theories around the factors influencing the determination of approaches and strategies of DevOps implementation. A comparison

of the theories and findings with literature led to the development of an adaptive model for DevOps implementation in organisations. This model was evaluated with an expert focus group. Using the model, we assessed the DevOps implementation of the organisations, identified gaps in their processes, and recommended actions to further improve their agility. The practitioners suggested a file structure to document the application of the model. The study thus concludes with the creation of a repository that reflects the model and is aimed at enabling a practical implementation of it.

1.5 List of Publication

The following papers are based on findings from this research:

- Macarthy, R. W., and Bass, J. M. (2020). An empirical taxonomy of DevOps in practice. In 2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA) (pp. 221-228). IEEE (Macarthy & Bass, 2020)
- Macarthy, R. W., and Bass, J. M. (2021). The Role of Skillset in the Determination of DevOps implementation Strategy. In 2021 IEEE/ACM Joint 15th International Conference on Software and System Processes (ICSSP) and 16th ACM/IEEE International Conference on Global Software Engineering (ICGSE) (pp. 50-60). IEEE (Macarthy & Bass, 2021)

1.6 Structure of Thesis

The rest of this thesis is organised as follows:

Chapter 2 – Literature Review gives an in-depth description of related concepts. Focused on the coordination of work between development and IT operations teams, the chapter describes agile software methodologies, continuous practices, tailoring, and the state-of-the-art of DevOps.

Chapter 3: Research Design presents the philosophical stance of the study and provides a detailed description of the research methodology followed.

Chapter 4: Taxonomy of DevOps Implementation in Practice, Benefits, and Challenges is a detailed presentation of the finding from phase 1 of this study, which is exploratory. It provides a thorough examination of practitioners' perception of DevOps.

Chapter 5: Strategies for DevOps Implementation. The Roles of Skillset and Automation describe findings from phase 2. The chapter provides an extensive investigation of DevOps implementation in organisations and the factors that influence the strategies and approaches taken.

Chapter 6: Model Creation - This chapter contains the creation process and description of a situational DevOps strategy engineering model. The model is based on findings presented in chapters 4 and 5, and literature.

Chapter 7: DevOps Strategy Engineering Model Evaluation is an expert Focus Group engagement to discuss the model. The chapter serves as an insight into practitioners' perceptions and describes the physical instantiation of the model created based on feedback from the participants.

Chapter 8: Discussion critically explores the study findings and answers the research questions. The chapter also presents the contribution to theory and practice. The limitations of the study are also discussed.

Chapter 9: Conclusion gives a concise summary and contributions of the thesis. A reflection on the entire Ph.D. is also given. The chapter concludes with useful directions for further research.

Chapter 2

Literature review

2.1 Introduction

The purpose of this chapter is to acquaint the reader with existing literature on efforts toward software development process improvements. This is evidence that the study is built on an extensive review of the literature and a good grasp of the research area, as required of any meaningful research (Boote & Beile, 2005). The presented literature was retrieved from various databases based on search criteria such as “software development methods”, “agile methods”, “software development improvement strategies”, “DevOps”, “work coordination in software development” etc. The databases include academic materials like books, reports, conference proceedings, journals, and articles. A rigorous analysis of the literature herein presented is the foundation of the research gap identification.

The chapter begins with an overview of the socio-technical nature of software development and methodologies for work coordination of development activities. Continuous integration is explored next, as an emerging development practice. Thereafter, the concept of tailoring software development methodologies is investigated, which narrows down to method engineering and then to situational method engineering. IT operations and continuous delivery is presented next. Finally, the coordination of work between development and IT operations is examined, leading to DevOps as a development to deployment coordination strategy.

2.2 Software Development Methodologies – Coordination Strategies for Development Activities

The evolution of software development methodologies can be viewed as consequent on the quest for optimised coordination of work among the actors in the process, and standardisation. Its continual improvement is aimed at the faster delivery of quality software. Beecham et al. (2010) describes coordination as specific activities related to identifying common goals and objectives, agreeing, allocating, and planning tasks among distributed teams. The complexity of computer-related systems, however, continues to increase alongside changing technology and requirements for such systems. Consequently, the degree of unpredictability of the software development process rises exponentially (Schwaber & Beedle, 2002) pushing the goalpost of optimised coordination even further away.

2.2.1 Socio-Technical Nature of Software Development

The available literature on improving the process of software development and delivery is extensive. While some studies have investigated the technological aspects of the delivery process (Haghighatkhah et al., 2018; Kersten, 2018b,a,c; Mäkinen et al., 2016; Mårtensson et al., 2018; Shahin et al., 2017; Ståhl et al., 2017), others examined the processes and social interactions (Ali et al., 2016; J. Highsmith & Cockburn, 2001; Dennehy & Conboy, 2017; Vlietland & van Vliet, 2015; Hummel et al., 2015). Debois (2008) draws attention to three layers of agile infrastructure: technical, projects, and operations. He identified the challenges in these layers and proposed mitigation strategies, one of which is the need to improve collaboration between projects and operations. The Agile Manifesto (Beck et al., 2001) emphasises the crucial role played by human interactions in software development and delivery. The use of tools to improve software development has also been widely studied. Building on Humble & Farley's (2010) recommendation of automation on the software deployment pipeline, Mäkinen et al. (2016) researched the state of practice in Finnish software companies to investigate the connection between tooling and continuous delivery. They concluded that organisations with less manual processes delivered software value faster. Their work is similar to the study done by Ståhl & Bosch (2013), as both focused on tooling in continuous practices implementation. However, while Ståhl & Bosch (2013) focused on continuous integration, their study extends to continuous deployment. Also, Ståhl et al. (2017) pointed out the significance of traceability for the successful execution of large-scale integration and continuous delivery and presented Eiffel, a traceability tool (Continuous practices are described later in this chapter). On the flip side, Kromhout (2017) stated that while tools are important, they do not by themselves solve complex socio_technical challenges like those seen in software development. They agree with Conway's law that any system designed by an organisation will reflect the organisation's communication structure. The study emphasises the importance of human communication and interactions of activities in the software delivery process. In the same vein, the culture of an organisation is said to be a determinant factor in the extent of its adoption of software development practices (Leidner & Kayworth, 2006; Fruhling & Tarrell, 2008; Cao et al., 2009). For instance, Gupta et al. (2019) explained the impact of IT department cultures on software development practices using the four Competing Values Model (CVM) cultural constructs: hierarchical, rational, group, and development. They tested the relationship between the cultural forms and two groups of agile practices: social and technical.

2.2.2 Agile Software Development

Agile software development is a philosophy of iterative and incremental development that has emerged to address the rapidly changing software environment

(Schwaber & Beedle, 2002) and manage complexity at the development phases of software projects (Lei et al., 2017). J. A. Highsmith & Highsmith (2002) describes it as a socio-technical process that emphasises short-cycled iterative social interaction among the different individuals and teams involved in the software development process. Following its advent, variant methodologies such as eXtreme Programming (XP), Scrum, Kanban, Lean software development, and Dynamic system development method (DSDM) have emerged, which align with the principles of the Agile manifesto (Beck, 2000) on varying levels (Dingsøyr et al., 2012; Lwakatare et al., 2016). Common themes span these methods like "let the architecture emerge", "deliver quickly and often", "empower team", "build for change", and "frequent customer input stem from the requirement-changing premise" (Bass, 2016; Dove & Turkington, 2008; Dingsøyr & Lassenius, 2016). Iteration is also a universal practice for all agile software development approaches. It "produces a working portion of the final product"(Dybå & Dingsøyr, 2008).This depiction agrees with similar literature (Boehm & Turner, 2004; Cockburn, 2004; J. A. Highsmith & Highsmith, 2002). According to the Agile manifesto and some agile methods, teams should be self-organising and cross-functional(Gutierrez et al., 2018). (Erickson et al., 2005) define agility as follows:

Agility means to strip away as much of the heaviness, commonly associated with the traditional software-development methodologies, as possible to promote quick response to changing environments, changes in user requirements, accelerated project deadlines, and the like. Previous research has categorised agile practices into social and technical categories (Chow & Cao, 2008)(McHugh et al., 2011). Hummel et al. (2015) advocated for the constructs "social agile practices", which they define as practices that aid collaboration and social interactions, and "technical agile practices", which refers to the coding and test-related activities in software development. Communication is promoted in Agile methods and is predicated on the interconnection of activities in the delivery pipeline. Although agile principles promote the idea of coordination and collaboration, especially between developers and users (Yu & Petter, 2014), it does not have prescribed methods of achieving it (Dingsøyr et al., 2012). Also, most of its principles like more frequent delivery are focused more on development process (Debois, 2008).

Scrum, Extreme Programming (XP), Lean software development, and Dynamic systems development method (DSDM) are described in the following sections. Agile roles, ceremonies, and artefacts come thereafter. This would pertain mainly to Scrum and XP, as they are the most widely adopted agile methods (Pikkarainen et al., 2008).

2.2.2.1 Scrum

Scrum, as described by Schwaber & Beedle (2002), is an approach that employs an empirical process control model to manage the software development process, and progressively adapts to changing circumstances. Scrum accentuates the need for active participation of customers, users, management, and developers in the setting of goals and evaluation of work at specific intervals to achieve business value at a faster pace (Schwaber & Beedle, 2002; Vlietland & van Vliet, 2015). Short development iterations in Scrum (sprints) enable quick feedback and consequent actions. Sprints are usually thirty-day (30) time-boxes of specific features development and delivery. Scrum is said to be a common Agile method in practice (Lei et al., 2017; Sverrisdottir et al., 2014). Scrum teams are self-organising, cross-functional teams responsible for the development and delivery of product features (Schwaber, 2004; Bass, 2014; Lei et al., 2017). The team is enabled by the structure of ceremonies and artefacts. Scrum of Scrums describes a team of the leaders of other Scrum teams.

2.2.2.2 Extreme Programming

Beck (2000) describes Extreme Programming (eXtreme Programming (XP)) as “an attempt to reconcile humanity and productivity” and a “social change” that encourages goal-oriented team spirit with its values, principles and practices. They assert that these make XP highly adaptable to the complexity and changes associated with software development. XP seeks to optimise software development at a lower cost, with reduced defects and higher return on investment by prescribing best practices for software development. Iyawa et al. (2016) claim that a deep level of interaction between customers and software developers is enabled by XP.

One of XP's most prominent practices is pair programming. Proponents of pair programming assert that it improves code quality (Beck, 2000). Meyer (2018) however claims that "that pair programming has no significant advantage or disadvantage compared to other techniques such as code inspection." He suggests that it be used on occasion. Other practices are test-first programming, whole team, continuous integration, ten-minute build, incremental deployment, etc.

2.2.2.3 Lean Software Development

Lean software development (Poppendieck & Poppendieck, 2003) is an adaptation of Lean thinking in manufacturing (Ohno, 1988), from which the principles of waste elimination, learning amplification, deferring commitment, delivering as fast as possible, team empowerment, building integrity, and see the whole were derived. The focus of this approach is value creation for the customer, providing quality codes and minimal cost (Yadav et al., 2018). Therefore, any effort that will not be paid for by the customer is considered a waste. The authors translated software waste to be: partially done work, extra processes, extra features, task switching, waiting, motion and defects. To amplify learning and build in integrity, Poppendieck & Poppendieck (2003) advocates for five actions: variability in development in the creation of appropriate solutions, short repeated development cycles, early feedback, iteration, and refactoring. Lean suggests an iteration that is time-bound, long enough to support meaningful value to the customer, and provide frequent feedback to the team. Like most agile methods, Lean advocates common ownership of code to facilitate synchronisation. Value stream mapping is used in Lean software development to provide insight into the internal working of the process/system and expose waste. The approach has also been employed in other investigations (Mujtaba et al., 2010).

2.2.2.4 Kanban

The Kanban software development approach is founded on the just-in-time philosophy. Kanban specifies what work is to be done at specific times by prioritising tasks, defining workflows, and lead time to delivery (Anderson, 2010). This approach centres on the use of a Kanban board to keep track of the progress of tasks and identify bottlenecks in the development process. The basic principles of Kanban are limit work in progress (Work in Progress (WIP)), pull value through the development process, process visibility, fixed backlog, and embedded. Kanban promises benefits like planning flexibility, shortened time cycles due to practices like code review and mentoring, visual metrics (use of control charts, and cumulative flow diagrams to track work efficiency (Abrahamsson et al., 2009; Dybå & Dingsøyr, 2008). Anderson (2010) stated that “Kanban has been shown to improve customer satisfaction through regular, dependable, high-quality releases of valuable software”. Although there are no explicit artefacts in Kanban methodology, backlogs, graphs, reports, and the Kanban cards are resultant artefacts of the method. Unlike Scrum, Kanban has no fixed sprints or cycle time and there are no defined roles or ceremonies. Kanban and Continuous Delivery complement each other as both focus on just-in-time value delivery(Lei et al., 2017).

2.2.2.5 Dynamic Software Development Method (Dynamic System Development Method (DSDM))

DSDM is described as an interactive and incremental agile framework for project delivery. The approach divides projects into three phases: pre-project, project life-cycle, and post-project. DSDM is guided by nine (9) underlining principles: frequent delivery, team empowerment, user participation, just-in-time development, iterative and incremental development, roll-backs, fixed initial high-level scope, continuous testing, and efficient and effective communication (Stapleton, 2003). Core techniques used in DSDM are timeboxing, configuration management, modeling, and MoSCoW (Must have, Should have, Could have, and Won’t have). DSDM prescribes a lot of roles for effective project management. They include the Executive sponsor, Visionary, Project Manager, Technical Coordinator, Team Lead, Solution Developer, Solution tester, etc.

2.2.2.6 Agile Roles

Three major roles are defined in scrum: the Scrum Master (who interfaces between Management, customers, and developers), the Product Owner (who is responsible for the "Product Backlog"), and Scrum teams. XP (Beck, 2000), on the other hand, defines roles such as testers, interactive designers, architects, project managers, product managers, executives, technical writers, users, programmers, etc.

2.2.2.7 Agile Ceremonies

Agile ceremonies are meetings with specific time frames. They are aimed at helpings teams plan, keep track, and update stakeholders. Scrum observes daily scrum where team members discuss what was achieved the day before, impediments faced, and what is to be achieved before the next meeting. This is similar to the Daily stand-up meeting in XP. A sprint planning meeting is held before any sprint in Scrum, and a sprint review meeting is held at the end of the sprint to evaluate the performance and deliverable of the Scrum team in the previous sprint (Lei et al., 2017; Schwaber & Beedle, 2002). Similar meetings such as weekly and quarterly meetings are also observed in XP.

2.2.2.8 Agile Artefacts

Agile artefacts are the information needed for the development of a product, or the product itself. Product Backlog, Sprint Backlogs, and Product increments. Product Backlog is a list of all conceivable features requested for a product, organised in order of priority by the Product Owner. Sprint Backlog contains items or features to be developed in a sprint. Product increments are new features and increments added to the product at the end of each sprint (Schwaber & Beedle, 2002; Lei et al., 2017). Bass (2016)identified "25 artefacts, organised into five categories: feature, sprint, release, product and corporate governance", which enabled consistency in Large-scale offshore software development programmes. Artefacts in XP include user story cards, quarterly themes, acceptance tests, release plans, iteration plans, task cards, design, unit test cases, and customer & developer communication records (Beck, 2000).

2.2.3 Continuous Integration - an Emerging Software Development Practice

To further improve the software development process, continuous practices are emerging. One of such is continuous integration. Continuous integration (CI) (Beck, 1999; Duvall et al., 2007) is a software development practice built on automation (Fowler & Foemmel, 2006). It encourages the committing of changes made to codes to the version control repository at least once a day. The CI server polls from the version control repository to identify changes made, builds, integrates, and inspects the whole software for conformity to rules and error checking (Duvall et al., 2007), to safeguard against broken builds. Continuous integration ensures the integrity of the software is maintained at any point in time and feedback is received as soon as possible. Software becomes more undeployable as the time spent to get it to a production-like environment increase (Duvall et al., 2007).

As a practice, continuous integration is independent of any tool (Fowler & Foemmel, 2006). It complements other development practices like developer testing, coding standard adherence, refactoring, small releases, and collective ownership. Values of CI include reduction in risk and repetitive manual processes, continual generation of deployable software (Fowler & Foemmel, 2006; Goodman & Elbaz, 2008), project visibility (Fowler & Foemmel, 2006), increased developer productivity (Goodman & Elbaz, 2008) and greater confidence in the product. Regression testing is a large part of CI, as it ensures that changes checked in do not negatively impact previously verified functionalities (Engström & Runeson, 2010; Haghighatkhah et al., 2018). There is no standard for the implementation of CI in practice. Consequently, organisations achieve dissimilar results from their execution (Ståhl & Bosch, 2013). Ståhl & Bosch (2014) isolated the consequences of CI implementation to adopted variants of it. There are however some constraints to adopting CI. Organisations see increased overhead in maintaining the Continuous Integration (CI) system, frequent change, too many failed builds, and additional hardware/software costs. Some see the activities being performed in CI as developers' basic responsibilities.

2.3 Tailoring Software Development Practices

As generally accepted in information systems development (Information Systems Development (ISD)), there is no one-size-fits-all approach to creating and delivering software systems (van Slooten & Brinkkemper, 1993; Henderson-Sellers & Gonzalez-Perez, 2005). Information systems researchers have argued on the practical direct application of prescribed methodologies (Avison, 1996; Baddoo & Hall, 2003; Glass, 2003). The appropriateness and suitability of accessing a suite of methodologies rather than insisting on one for all projects in an organisation is argued. (Avison, 1996) suggested the possibility of an era of software development without "control, standards or training" emerging from the viewpoint of the inability of generic methodologies to address important social, political, and organisational factors.

Ginsberg & Quinn (1995) defines tailoring as "the act of adjusting the definition and/or particularising the terms of a general description to derive a description applicable to an alternate (less general) environment". Process tailoring is well established in the literature. A systematic literature review (Kalus & Kuhrmann, 2013) identified 49 criteria whose "relevance to particular projects (they claim) can significantly influence the resulting software process". This 'relevance', however, seems subjective from the study. They also formulated 20 actions considered to be standard measures to address project situations and related them to the tailoring criteria identified. For tailoring to apply, processes and practices must be well-defined (Ginsberg & Quinn, 1995). Kalus & Kuhrmann (2013) conclude that the factors affecting the tailoring of software development practices are well understood. However, the consequences of the criteria remain abstract and should be interpreted on a project basis.

Cameron & Quinn (2011) pointed out that to achieve an enduring improvement in organisational performance, processes and procedures must be integrated with organisational change. The adoption of any Agile method or continuous practice is perceivable as an organisational change. However, processes and procedures of similar Agile adoption differ in practice. Furthermore, Agile methods were intended for small, self-managing, collocated teams. As such, its application to large-scale projects requires tailoring (Bass, 2016; Dingsøyr et al., 2017). Tailoring of the practices and roles by organisations results in differing outcomes even when the same methodologies are adopted. Ståhl & Bosch (2014) showed how the variant implementation and interpretation of CI in practice can cause a disparity in the realisation of its adoption goals. This perceived variability agrees with the study by Bass & Haxby (2019), which identified three groups of activities that must be managed by the product owners in a large-scale SD project: scale, distance, and governance.

2.3.1 Method Engineering

Method engineering was introduced by Bergstra et al. (1985) and was referred to as “Methodology engineering” by Kumar & Welke (1992). Brinkkemper (1996) and van Slooten & Brinkkemper (1993) however insisted that the term “method engineering” is more appropriate. This is more generally accepted. Brinkkemper (1996) defines method engineering as “the engineering discipline to design, construct, and adapt methods, techniques, and tools for the development of information systems”. Harmsen et al. (1994) defines a method as “an approach to perform systems development project, based on a specific way of thinking, consisting of directions and rules, structured systematically in development activities with corresponding development products.” This agrees with Brinkkemper (1996) definition which says “an information systems engineering method is an integrated collection of procedures, techniques, product descriptions, and tools for effective, efficient and consistent support of the Information Systems (IS) engineering process.”

2.3.2 Situational Method Engineering

SME emerged from ISD process research to address the need to provide method engineers and organisations with structured flexibility to “tailor” software development methods according to their unique context (Bucher et al., 2007). According to Brinkkemper (1996), “a situational method is an information system development method tuned to the situation of the project at hand. Similarly, Henderson-Sellers et al. (2014) described SME as “an alternative approach to software development method based on local conditions”. Several terms such as “project environment” (Brinkkemper et al., 1998; Harmsen et al., 1994), “project situation”(Karlsson & Ågerfalk, 2004), “reference context” (Baumoel, 2005), “situation” (Rolland, 1998) are used in reference to “situation” in SME.

The importance of situational characteristics and factors is well understood in literature (Brinkkemper, 1996; Harmsen et al., 1997; Rolland & Prakash, 1996), and has been the focus of many SME research. All the studies I reviewed agree that the characteristics of the relevant development situation need to be explained and generic methods should be adapted according to situational characteristics.

Situational characteristics are noted to be considered at levels such as organisation, process, or projects (Hoppenbrouwers et al., 2011). Bucher et al. (2007) proposed a set of extensions to the method engineering meta-model to make allowance for a differentiation between “context” and “project type” in Situational Method Engineering (SME) approaches, and grouped the adaptation mechanism of method engineering into 2:

1. Situational method configuration – which is the tailoring of a single base

method to a specific development situation.

2. Situational method composition – which involves the “selection and orchestration of artefacts fragments” from several methods, to construct a new situational approach based on a specific project situation. This seems to be the more generally accepted approach to SME.

Exploring the situational method composition seems more suited for my work. Based on Brinkkemper (1996)and Harmsen et al. (1994), the composition process is divided into 3 phases following well-defined construction principles:

1. Identifying situational characteristics.
2. Decomposing generic artifacts into artifacts fragments
3. Composing artifacts fragments into situational method

Some other studies provide guidance for identifying situational characteristics (Situational Characteristics (SC)). The application of the MADIS modeling framework to characterise both the problem situation and the artefacts fragments (Essink, 1986) was proposed by (Punter & Lemmen, 1996). This essentially viewed the software development process from various levels of abstraction: object system modeling, conceptual IS modeling, data system modeling, implementation modeling, etc at similar domains. Rolland & Prakash (1996) identified SC by problem domain and subject area, exploring factors like complexity and risk within the problem domain. It is not clearly explained in the study why these factors were chosen. The use of 17 contingency factors to describe the environment of the method application was proposed by Van de Ven et al. (1976). Brinkkemper (1996) insists that “engineering a situational method requires standardised building blocks and guidelines, so-called meta-methods to assemble these building blocks.” Figure 2.1 shows the configuration process for SME proposed by Brinkkemper (1996) to guide the assembly of the “building blocks” into situational methods.

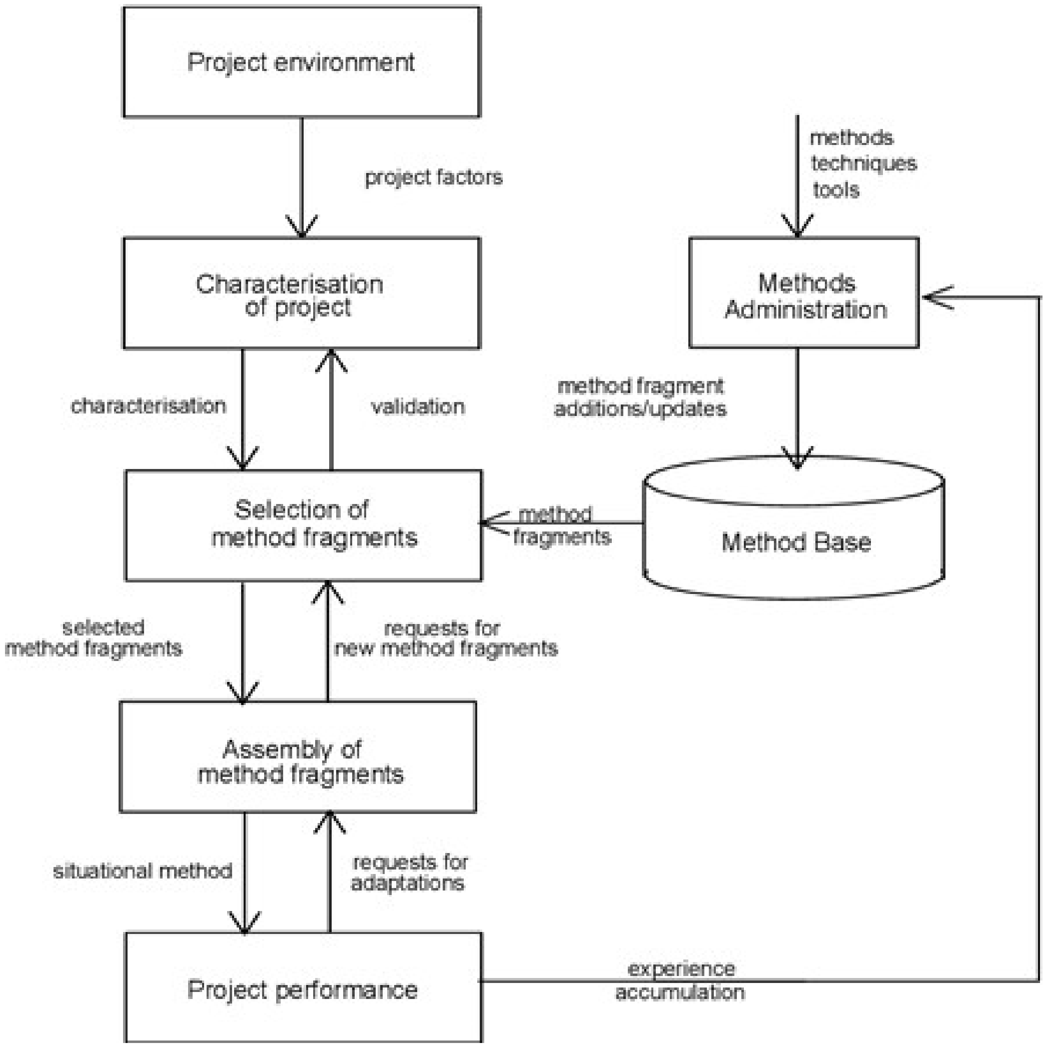


Figure 2.1: The configuration process for SME

Fig 2.2 is Henderson-Sellers et al. (2014) description of SME construction. The method engineer selects method parts from a method base ("which are all conformant to an element in the metamodel") and constructs an organisationally specific methodology using proven construction guidelines and situational factors. They stated that the situational characteristics are "integrated with the method parts extracted from the method base".
Comparing Fig 2.1 and Fig 2.2, similar notions such as "method base", and "method parts" etc are found.

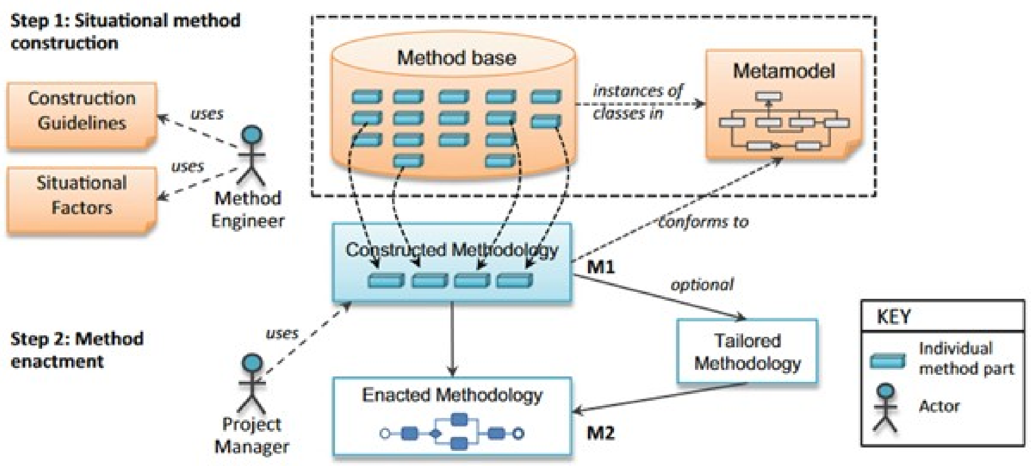


Figure 2.2: SME construction

Method Base is described as a "repository" of method parts (Brinkkemper, 1996; Harmsen et al., 1997; Rolland, 1998). According to Henderson-Sellers et al. (2014), method parts are "small portions of a methodology, either a methodology that already exists or a methodology to be formed", that describe in detail a specific task, technique, work product, etc.. They insist on the standardisation and conformity of each part to "a higher-level definition given in a meta-model". Various authors describe method parts with terms such as method fragments, method chunks, and method components.

Method Fragments: Harmsen et al. (1994) coined the term "method fragment" and explained it as "...a description of an IS engineering method or a coherent part thereof". The term was popularised by Brinkkemper (1996). The description agrees with that of Ter Hofstede & Verhoef (1997) – "a coherent part of a metamodel, which may cover any of the modeling dimensions at any level of granularity". According to Henderson-Sellers et al. (2014), the fragmentation and structure of the method knowledge base make it straightforward for more fragments to be added to the method base, arising from specific organisational tailoring.

Method Chunk: Some authors explore method parts at a higher level of granularity and refer to them as method chunks (Rolland & Prakash, 1996; Rolland, 1998). This is described as "the combination of a process part plus a product part.

Method Component: Ro¨stlinger A (1994) viewed methods as comprising transferable and reusable components. Method components consist of “descriptions for ways of working (a process), notion and concepts. Here, a process is the description of the rules and order of action performance. Notion refers to the rules for documentation, including syntax and diagrammatic representations. The basic categories of processes and notions are referred to as concepts.

Metamodel: A metamodel is a detailed description of a specific task, technique, work product, etc. For example, how to draw a use case diagram (Henderson-Sellers et al., 2014). This in itself is a model. ISO (2014) explores meta-modeling in the following domain areas:

1. Work units: a description of the work required to obtain the expected system.
2. Work product: This describes the artefacts to be used or created to achieve the expected system.
3. Producers: This describes the roles, teams, and tools required to carry out the work units and produce or make use of the work products.
4. Stages: A description of the phases of interaction between work units, work products, and producers in the delivery value chain.
5. Model units: Building blocks for the construction of work units.

According to ISO (2014), the International Standard to define Software Engineering Meta-model for Development Methodologies (SEMDM), “combines key advantages of other metamodeling approaches (Firesmith & Henderson-Sellers, 2002; Gonzalez-Perez & Henderson-Sellers, 2005; Hawryszkiewycz, 2002; Schuppenies & Steinhauer, 2002), allowing the seamless integration of process, modeling, and people aspects of methodologies. Thus, the standard facilitates methodology assembly from a repository of method fragments and the interaction between the various domain areas (Gonzalez-Perez, 2007), unlike Object Management Group (OMG)’s model Schuppenies & Steinhauer (2002) representation of subjects strictly by "instance-of" relationships (usually organised in meta-levels). Figure 3 shows Gonzalez-Perez (2007) representation of the structure of International Organisation for Standardisation (ISO)/IEC 24744.

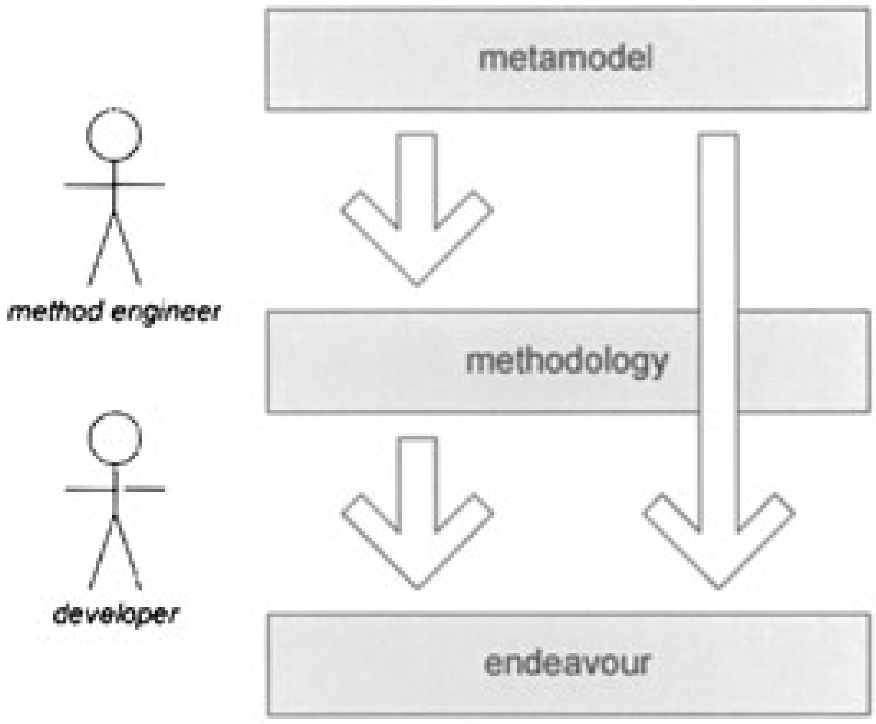


Figure 2.3: ISO Model

They describe three domains: metamodel, methodology, and endeavour. Rather than the "instance-of" relationship, metamodel here is shown to be related to both methodology as well as endeavour. Consequently, endeavours such as work products can be used to construct method fragments and populate repositories.

Brinkkemper (1996) suggested a framework for hierarchical meta-modeling from 3 orthogonal dimensions: perspective, abstraction, and granularity, for the classification of method fragments.

Table 1.1 summarises Aydin (2007) key notions of SME.

Table 2.1: SME notions(Aydin, 2007)

Notion	Basic View	Extension
Situation	Characterised by several factors that influence or are being influenced by a method fragment	The limited parts of reality that the agency perceive, reason about, and live in
Context	Described in terms of aspects of collectives in the process	Dynamic interplays among collectives of work practice as situated and characterised by the agency
Agency	Adheres to the enactment of proposed fragment in the work practice	Interlays among fragments with a certain intention in and for the context
Method fragment	Description of a methodical artefact or any coherent part thereof	Comes into play with the agency in the context when structuring one's thinking and actions

2.4 Information Technology (IT) Operations

The delivery of a working software application to the intended user, and its efficient management is the end product of any software development effort (Hall et al., 1999). This delivery is made possible through the instrument of IT operations. IT operations include activities around the release of software such as installation and configuration, and maintenance activities like monitoring, updating, reconfiguration, redeploying, and decommissioning of the software (Cao et al., 2009). Essentially, the development team hands over the software to the operations team at this point, which is a huge ceremony. For many organisations, it is a long, tiring process, shrouded by unpredictability. This puts all the actors under immense pressure and results in blame games when there are failures.

Several studies propose frameworks and approaches to improve the software delivery process. Hall et al. (1999) suggested a Software Dock framework that "creates a distributed, agent-based deployment framework to support the ongoing cooperation and negotiation among software producers themselves and among software producers and software consumers". This framework was at developers providing their customers with high-level deployment services. Talwar et al. (2005) discuss service delivery approaches such as script-based, language-based, and model-based, and the trade-off between initial cost and repeated costs in relation to automation. To reduce the complexity of deployments caused by dependencies and interaction between the product being installed, virtualization is employed (Dearle, 2007). Virtualization is creating a level of abstraction away from the physical systems and running several virtual systems on a single hardware. According to Schumate (2004), service deployment in such an environment consists of 5 steps: creation, generalisation, testing, distribution, and update. Despite these strides towards a more simplified deployment process, service delivery continued to be a huge deal for organisations.

2.4.1 Continuous Delivery - an Emerging IT Operation Practice

Continuous delivery (Continuous Delivery (CD)) is a practice that ensures software is always kept in a releasable state (Chen, 2015; Proulx et al., 2018; Siqueira et al., 2018). It seeks to answer the question: how are the activities in software development coordinated to achieve speed, efficiency, and reliability in software delivery? (Humble & Farley, 2010; Rodríguez et al., 2017; Leppänen et al., 2015). This concept shares similar ideas and philosophy with Lean development. Perceived benefits include faster time to market, team empowerment, error reduction, and deployment flexibility (Chen, 2017; Leppänen et al., 2015). Continuous delivery stands on the pillars of automation and feedback (Humble & Farley, 2010). Its principles include creating an auditable release process, frequent releases, automated testing, and keeping everything in version control- to provide access to all versions of the stored file and its meta-data (Humble & Farley, 2010). However, challenges in Continuous Delivery implementation such as resistance to change, manual and non-functional testing, process alignment, domain constraints, and tooling have been identified in both literature and practice (Chen, 2015, 2017; Leppänen et al., 2015).Chen (2017) recommended the establishment of dedicated teams with multi-disciplinary members as a mitigating strategy for some of the above-mentioned challenges: an alignment with the DevOps philosophy.

Of importance in Continuous Delivery is configuration management – “the process by which all artifacts to your project, and the relationships between them, are stored, retrieved, uniquely identified, and modified” (Humble & Farley, 2010). Configuration management is to facilitate environment reproduction, traceability of changes in any environment, and visibility. The end-to-end process from a commit to the release of a feature/application to production is termed the deployment pipeline. A large portion of waste in the software development process stems from movement through testing and operations teams. Visibility on the deployment pipeline provides an insight into the bottlenecks of the delivery process (Chen, 2017), and is achieved using value stream mapping (Humble & Farley, 2010).

The implementation of continuous practices is seemingly a promising way to improve software delivery. Coordination and communication among team members have been identified as key factors in the successful adoption of Continuous practices (Shahin et al., 2017). While continuous practices promise faster delivery of value to customers, the impact of the processes and interactions within the value stream cannot be undermined. Its success is heavily dependent on the seamless collaboration of the teams working on various aspects of the delivery process and an efficient product flow from one stage to another (Mäkinen et al., 2016).

2.5 Coordination of Work between Development and IT Operations in Software Development

Coordination has been defined as “the integration or linking together of different parts of an organisation to accomplish a collective set of tasks” (Van de Ven et al., 1976), and “managing dependencies between activities” (Malone & Crowston, 1994). Dependencies exist in multi-team projects and require synchronisation, feedback, information-sharing, and conflict resolution (Dingsøyr et al., 2017; Galbraith, 1973), the degree of which is directly proportional to the level of complexity and uncertainty of the project (Dietrich et al., 2013; Kraut & Streeter, 1995) argued that “coordination becomes much more difficult as project size and complexity increases," and identified deficiency in coordination as the cause of most problems in software development. The study established scale, uncertainty, and interdependence as some characteristics influencing coordination. According to Noll et al. (2011), "the key barriers to collaboration in global software development are geographic, temporal, cultural, and linguistic distance". The authors proposed "site visits, synchronous communication technology, and knowledge sharing infrastructure to capture implicit knowledge and make it explicit" as solutions. Hoegl & Weinkauf (2005) emphasised the importance of managing task inter-dependencies as the quality, performance, and development time of multi-team projects is highly reliant on the "coordination pattern." Software development has applied approaches like technical tools, modularisation, and formal procedures like version control to enable coordination. These techniques contributed to productivity but did not sufficiently address the challenge of coordination (Kraut & Streeter, 1995).

Coordination is usually attained through mechanisms such as meetings, workshops, phone calls, and IT communication tools (Dietrich et al., 2013). The study investigated coordination in multi-team projects through identification of mechanisms, modes, and patterns in practice. Prior collaboration studies to theirs were either on single-team projects or organisations themselves, which significantly differs from multi-team projects. While organisational coordination can be achieved through such things as validated procedures, written policies, and job descriptions (Galbraith, 1973), the same cannot be said of more transients structures like projects with flexible requirements and team-members (Dietrich et al., 2013; Kerzner, 1998). Dietrich et al. (2013) discovered three patterns of coordination: centralised, decentralised, and balanced patterns, expressed in either group or individual mode of personal coordination, or impersonal coordination mode. These modes were previously identified by Kraut & Streeter (1995) and (Van de Ven et al., 1976). The study also shows significant impact of coordination on the outcome of "information sharing, workflow fluency between teams, efficiency of projects and learning". Determinant factors of coordination pattern selection identified in organisations are innovation (Dietrich et al., 2013), technological and workflow correlations between teams (Van de Ven et al., 1976; Andres & Zmud, 2002), and task "decomposability" (Nidumolu, 1996). (Van de Ven et al., 1976) found that a higher degree of uncertainty in projects forces the adoption of group mode of coordination. This was further established by (Dingsøyr et al., 2017) and agrees with (Kraut & Streeter, 1995) study which suggested informal, interpersonal communication under uncertain conditions. Dietrich et al. (2013) however considers uncertainty and complexity in the light of coordination patterns in projects and suggested that the decentralized pattern is better suited for projects with a high degree of uncertainty. An investigation by Dingsøyr & Lassenius (2016) into factors influencing the performance of co-located teams proposed such factors to include team collaboration and shared mental models which, they stated, is an important tool for understanding team collaboration patterns. Although the study alludes to the possible existence of such patterns, an in-depth explanation of the concept was not provided. Dingsøyr et al. (2017) also studied the usage of group coordination mode in large-scale software development projects that showed the transition between scheduled and unscheduled meetings. The focus of the study seemed to be limited to development teams. It did not also take the patterns of coordination between development and operations activities into consideration.

2.6 DevOps – A Development-to-Deployment Coordination Strategy

A recent attempt to handle coordination in software development is the introduction of DevOps (Len et al., 2015; Kim et al., 2016; Senapathi et al., 2018). DevOps (Development and Operations) is described as an emerging software engineering culture and philosophy that utilises cross-functional teams (development, operations, security & QA) to build test, and release software faster and more reliably through automation (Dyck et al., 2015; Kim et al., 2016; Senapathi et al., 2018). From literature, it is understood that DevOps seeks to bridge the gap between the conflicting priorities of the development and operations teams. According to Bass et al.(Len et al., 2013), "DevOps community advocates communication between the operations staff and the development staff as a means of ensuring that the developers understand the issues associated with operations." However, for Rowe & Marshall (2012), the actual gaps to be bridged are "disconnects between processes, measurements, technologies, and data." I view the above as a more granular description of the "gap" between development and deployment teams. Nybom et al. (2016) stated that three possible approaches to bridging the gap are: "Mix responsibilities: assign both development and operations responsibilities to all engineers, or Mix personnel: increase communication and collaboration between Dev and Ops, but keep existing roles differentiated, or Bridge team: create a separate DevOps team that functions as a bridge between Devs and Operations (Ops)".

In their study, López-Fernández et al. (2021) provides a taxonomy of DevOps team structures which they described as i) interdepartmental Dev & Ops collaboration - the temporary collaboration between development and operations teams for specific project, ii) Interdepartmental Dev-Ops team - the unification of development and operation teams, characterised by shared product ownership, iii) Boosted cross-functional DevOps team - using DevOps experts to "boost" development teams till they become DevOps teams themselves, and iv) Full cross-functional DevOps team - product teams with both development and operations skills. Their classification was based on factors such as Leadership from management, shared product ownership, collaboration frequency, organisational silos, cultural silos, and autonomy. They compare the teams structures and performance and conclude that "companies that implement the most mature team structures (i.e., iii and iv) achieve better software delivery performance indicators". (Leite et al., 2022) also carried out a study to understand the reasons behind the various DevOps structures adopted by organisations and identified 4 structures: segregated departments, collaborative departments, API-mediated departments, and single department. Their description is similar to that of López-Fernández et al. (2021). However, they provide causes of such structures like startups are inclined to single departments and the bid to overcome existing delivery bottlenecks leading to API-mediated departments.

The DevOps cycle is described as comprising eight main phases that encompass the phases of the software development life cycle (include reference). They include:

1. Plan: This comprises business value and requirements definition.
2. Code: The code phase involves the design and the creation of software code
3. Build: This involves configuration management and version control, and the use of automated tools for code compilation and packaging code for future production releases.
4. Test. The test phase involves continuous testing activities (manual or automated) to ensure optimal code quality.
5. Deploy. This phase can include activities and tools that help to move product releases into production.
6. Operate. The process of managing software in production.
7. Monitor. This involves the collection of relevant metrics and information about issues from specific software releases in production.

In practice, various tools are associated with each of these phases. An example of this is shown in Table 2-1. This table is not intended to be a representation of the DevOps tools ecosystem, as tools are also often classified for activities in the DevOps cycle such as continuous integration, continuous delivery, security, logging, configuration management, communication, etc., but is rather an example showing the multifariousness of DevOps tools ecosystem (Include reference).

2.6.1 The Nature of DevOps

DevOps is described as a software engineering culture and philosophy that utilises cross-functional teams, to build test, and release software faster and more reliably through automation. According to available literature, benefits derived from DevOps adoption include faster delivery, improved quality & security, and better collaboration (Nybom et al., 2016; Roche, 2013; Lwakatare et al., 2019). Senapathi et al. (2018) stated that "teams are happier and more engaged", and shared technical knowledge between operations and development teams increased collaboration between them after DevOps implementation. Erich et al. (2017) claim that existing literature shows no agreement on the definition or scope of DevOps, nor evidence of its effectiveness. Smeds et al. (2015) insist that the foremost views can be identified in the blogosphere: the view that DevOps is a cultural movement to facilitate rapid software development and deployment, and the argument that it is rather a job description requiring both development and IT operation skills (Nybom et al., 2016; Senapathi et al., 2018; Smeds et al., 2015). The former view seems more predominant in available peer-reviewed literature. Humble (2012) stated clearly in his blog post that "there's no such thing as a 'DevOps team'." He however explains that IT operations teams can sometimes be referred to as 'DevOps teams' when they fulfill some specific responsibilities such as the automation of deployment pipelines and provision of support for developers. Hüttermann (2012) also insisted that DevOps is neither a job description nor a department or a unit in an organisation. He stated that "DevOps describes practices that streamline the software delivery process, emphasising the learning by streamlining feedback from production to development and improving the cycle time." Willis (2010), agreeing with Hüttermann (2012) stance, identifies four significant characteristics of DevOps in a blog post as culture, automation, measurement, and sharing. This blog post was referenced in Nybom et al. (2016). However, the 2014 State of DevOps report shows a growing number of DevOps teams (Forsgren et al., 2014). The ambiguity and conflicts in the description of the concept have resulted in organisations taking different approaches to DevOps. Senapathi et al. (2018) identified four DevOps descriptions in literature: merging of development and operations for closer collaboration, development that supports operations, centralised group, and incentive alignment. Although the paper states the approach taken by the organisation under study, details of an actual implementation are scanty. Wahaballa et al. (2015) presented a road map for the standardisation of DevOps terminologies and practices. The argument that DevOps should be a job description is mainly seen in blog posts. However, none has been referenced here as I could not find peer-reviewed literature with such reference. In addition, the scope of DevOps differs in its adoption (Erich et al., 2017). While some organisations view DevOps as just "deployment automation" by select cross-functional teams, others like

International Business Machines (IBM) consider it to be “improved automation, integration, collaboration, and optimisation of development and operations” (Rowe & Marshall, 2012).

Although there are no prescribed ceremonies in DevOps, it advocates practices like continuous integration, continuous delivery, continuous deployment, automated testing, infrastructure-as-code, and automated releases. There are however challenges such as the lack of appropriate skill-sets while implementing DevOps. Other issues such as the fast-evolution of technology stack and tools as well as resistance to change have been identified in the adoption journey (Smeds et al., 2015; Nybom et al., 2016; Senapathi et al., 2018).

2.6.2 The Impact of DevOps

Beyond anecdotal evidence (Rembetsy & McDonnell, 2012) and survey data, however, empirical study of the impact of DevOps on the software value stream is scarce in literature (Erich et al., 2017). Senapathi et al. (Senapathi et al., 2018) carried out an exploratory case study to provide empirical evidence of the impact of DevOps adoption in a New Zealand organisation, where DevOps was perceived as “embedded Ops.” They found the employment of automation and cross-functional teams as facilitators of DevOps value delivery. Kersten (2018c) introduced a flow framework to create a value stream integration diagram which he said would give perspective to how DevOps worked in practice. The framework measured the flow velocity, efficiency, time, and flow load against the business results like value, costs, quality, and happiness. The diagram is said to provide a real-time summary of tools, their interconnection, and artefacts within a value stream. Kersten (2018b) classified flow items all flow items in the value stream to be features, defects, risk, and debt which he said abstracts a layer away from Kruchten (1995) positive/ negative versus visible/invisible quadrant. Flow techniques are employed by organisations to identify bottlenecks in their delivery process. Dennehy & Conboy (2017) investigated the adoption of flow techniques using Activity Theory and suggested research into the combination of flow techniques with Activity theory, which he asserts would give a richer perspective to "contradictions and congruencies" on the software development value stream. (Ali et al., 2016) combined flow technique with value stream mapping to identify waste. The study is however limited to the specification stage of the development process.

2.7 Tailoring in DevOps Implementation

Studies have established discrepancies in the implementation of DevOps in various organisations (Nybom et al., 2016). While tailoring is widely accepted and even advocated for in the software development industry (Bass & Haxby, 2019; Fitzgerald et al., 2006; Kalus & Kuhrmann, 2013), its results are unpredictable (Ståhl & Bosch, 2014). The concept of tailoring practices in DevOps is quite tricky. This may also be attributed to the ambiguity surrounding the definition of the concept itself. In addition, there are no prescribed practices for DevOps besides enablers such as automation and continuous practices. Two limitations to applying the tailoring criteria, applicable to Agile software development, identified in the literature to DevOps implementation are:

1. The tailoring criteria are applied uniquely to individual projects. DevOps implementation on the other hand is perceived as an organisation-wide change in the sense that the same structural or cultural change is expected across teams in an organisation.

2. Tailoring works on the principle of the variability of standards within each project. DevOps advocates for standardisation and automation of processes within organisations. Although fragments of various methods are used in DevOps, making the traditional tailoring of the software development process is a subset of the DevOps implementation pattern.

To address the points raised above, I extract the organisational level tailoring criteria from the tailoring criteria identified in literature (Kalus & Kuhrmann, 2013). This abstracts away from tailoring within projects to "tailoring" on an organisational level, to enable the evaluation of DevOps implementation within organisations. My research direction is informed by future work pointed out by Leite et al. (2019) and Wiedemann & Schulz (2017) who both insist that an in-depth study into organisations implementation of DevOps is critical to the successful adoption of the concept in the software industry.

Literature on DevOps adoption is mainly on DevOps capabilities, technological and cultural enablers, and the benefits and impediments to its adoption (Chen, 2017; Erich et al., 2017; Leite et al., 2019; Smeds et al., 2015). Amaro et al. (2022) conducted a Multivocal Literature Review to understand the relationship between capabilities and practice, and better map DevOps implementation. The study concludes that "capabilities" is a dynamically evolving aspect of DevOps. They stated team collaboration and communication as the most crucial ones.

Rodrigues et al. (2020) showed correlations between business strategies, business values, firm capabilities, and firm performance. They describe business strategies as capacity utilisation, rate of product/service innovation, modernisation, and automation of processes, efforts to achieve economies of scale, etc. Business values are presented as intermediate benefits derived by organisations such as support in decision making, quality of information, product/service quality, on-time delivery, etc. Firm capabilities, as mentioned in the study are relatable to the software development tailoring criteria identified in literature (Kalus & Kuhrmann, 2013), except that the latter takes into consideration projects peculiarities. I equate firm capabilities in my proposed study to tailoring criteria, extracting only organisational level characteristics. I describe this as the organisation context. This consist of organisation domain, organisation size, team size, team distribution, degree of innovation, domain knowledge, technology knowledge, tool knowledge, process knowledge, system architecture, communication, financial controlling, legal aspect, and culture. This research groups technology knowledge, tool knowledge, and process knowledge as a skillset in relation to DevOps implementation.

Seven skill categories required to set up DevOps teams: full-stack development, analysis, functional, decision-making, social, testing, and advisory skills - with 36 concrete skills were presented by Wiedmann and Wiesche. They also highlighted the combination of distinct development, operations, and management skills necessary to successfully work within such teams. The global skills and competency framework for a digital world (Skills Framework for the Information Age (SFIA)) describes an assembly of resources related to about 30 professional skills and competencies required for DevOps which they claim "reflects that successful approaches to DevOps need much more than simply deploying technical tools to automate development and operational IT tasks". They divided this into seven (7) levels of responsibility (Society, 2012). No literature, however, was found that provides an investigation into the relationship between skillsets and the approach an organisation adopts to implement DevOps.

The impact of DevOps practices on the software development process has also been studied by (Hummel et al., 2015; Perera et al., 2016). organisations are faced with numerous paths to DevOps implementation. Little is known about how to string together the wealth of information scattered in the literature, to successfully implement DevOps based on a specific organisational context. The question, "What informs organisations mode of implementation of DevOps?" remains largely unanswered (Leite et al., 2019; Luz et al., 2018, 2019) present a model for DevOps adoption, comprising of three (3) steps:

- In the first step, a company should disseminate that the goal of DevOps adoption is to establish a collaborative culture between development and operations teams.
- In the second step, a company should select and develop the most suitable enablers according to its context. The enablers are means commonly used to develop the collaborative culture and its concepts.
- In the third step, a company should check the outcomes of the DevOps adoption to verify the alignment with industrial practices and to explore them according to the company's needs.

The study, however, does not sufficiently provide answers to the following questions:

1. What factors would determine the selection of a "suitable enabler"?
2. What is the relationship between these factors and the enablers?

These queries are aligned with my research questions in the sense that I query the roles of elements of organisational context in the implementation of DevOps, not only regarding team topology but as it affects the strategy of implementation.

According to Wiedemann et al. (2019) "DevOps requires a custom solution for each organisation". Kromhout (2017) also emphasised the importance of paying more attention to the mode of DevOps implementation in organisations.

According to reports (Forsgren et al., 2014) its popularity is fast rising among software development practitioners. From the literature herein presented, substantive research exists around understanding the concept of DevOps and its practices. However, there is evidence that the multifaceted nature of available information on DevOps would be quite daunting for some who wish to embark on the journey (Senapathi et al., 2018; Riungu-Kalliosaari et al., 2016). Luz et al. (2019) claims that "It is still unclear how one could leverage such rich, yet scattered, information in an organised and structured way to properly adopt DevOps." This presents the transition to DevOps as a decision to be made with careful consideration. Surveys such as that conducted by Leite et al. (2019) show an increasing interest in research into DevOps. However, subjects such as the pathway to implementation and organisational factors influencing such decisions are still less studied than investigations into its nature and value derived from its adoption (Leite et al., 2019). They stated that while engineers need to learn how to re-engineer their systems and qualify themselves for DevOps-related positions, managers seek to know how to introduce DevOps to their organisations and how to assess the quality of already adopted DevOps practices. As there are no specific guidelines, changes to the operating culture can either be successful and yield the benefits promised by DevOps, or fails and be detrimental to the organisation's operational stability and much more. This indicates a need to critically examine the interactions between the activities and parts of DevOps implementation in organisations.

This research is motivated by a general lack of scholarly guidance in the determination of a suitable path to DevOps implementation. It attempts to organise information on the various components of DevOps to help practitioners better align their coordination activities and avoid pitfalls. The research will also investigate DevOps approaches and their determinants. This would provide substantial value to both understanding and implementing the concept.

2.8 Summary

The chapter presents available literature on software development methodologies, agile software development, and continuous practices. The concept of tailoring software development practices (including method engineering and situational method engineering) is herein introduced. The concept and nature of DevOps are explored. An in-depth analysis of the literature exposed the scarcity of knowledge on the suitable pathways to DevOps implementation. This gap is also identified by a recent DevOps survey (Leite et al., 2019), and forms the foundation of the study. The next chapter describes the methodologies used to carry out the study.

The research adopts an exploratory multi-case study approach to investigate DevOps implementation in industrial settings, based on interviews with practitioners. My data will be analysed using an approach informed by grounded theory, which would allow theories to emerge, grounded in the data (Glaser & Strauss, 1967; Glaser, 1992; Stol et al., 2016).

Chapter 3

Research Design

Research Design

3.1 Introduction

Research methodologies provide structured and progressive guidance, suitable for specific kinds of academic investigations. Research is thus substantiated by adherence to the prescribed guidelines of chosen methodologies and should be carried out systematically. The design of this research takes guidance from the research onion(**?**).

This chapter presents the research structure and the steps taken to answer the research questions. It also presents the justification for the suitability of the methodologies to achieve the aims and objectives of this research. Section 3.2 describes the methodological options available and the choices for this research. The research purpose and outcome are discussed in Section 3.3. This is followed by a full description of the research design including data collection and analysis, and the DevOps strategy model development and evaluation.

3.2 Research Model

A research model is the theoretical image of the object of study (Palvia et al., 2006). The determination of an effective research structure "involves interconnection and interaction among different design components" (Maxwell, 2012). Figure 3.1 is an adapted research onion showing possible choices for each layer of the research structure. The "methodological choice" layer shows the 3 basic types rather than the possible variations of them.

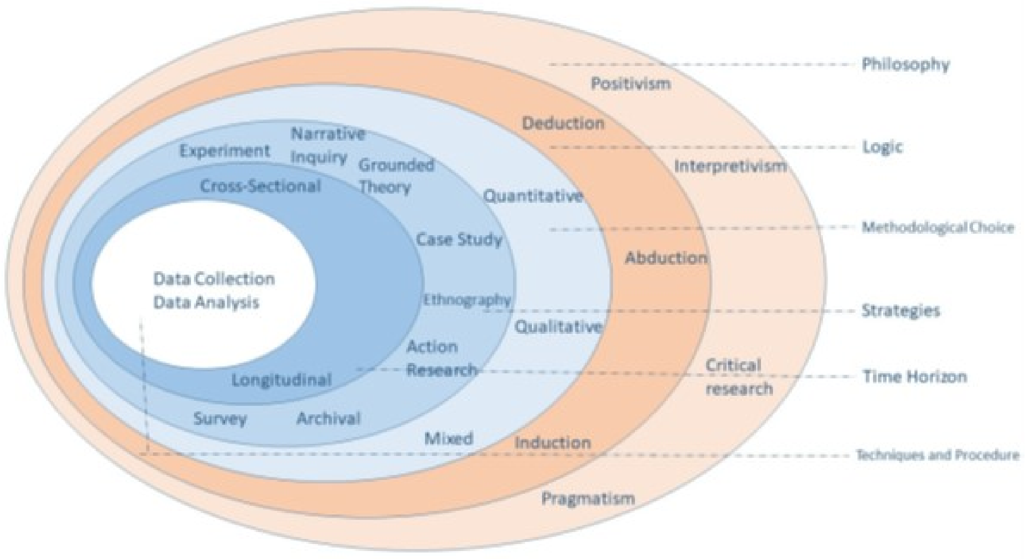


Figure 3.1: Adapted Research Onion (Saunders et al., 2019)

3.2.1 Research Philosophy

Research philosophies are "underlining assumptions of what constitutes valid research" and the appropriate method to employ. Four epistemological perspectives established in literature are positivism, interpretivism (or constructivist), critical research (or participatory), and pragmatism (Myers & Avison, 2002; Myers, 1999). **Positivism** assumes that the world is ordered and regular, that reality can be objectively measured and described using instruments (Myers & Avison, 2002). **Constructivism** (interpretivism) seeks to understand the world and events through human perceptions (Sekaran & Bougie, 2016). **Critical realism** aligns with positivism in the assumption of objective truth but argues against objective measurement (Sekaran & Bougie, 2016). **Pragmatism** believes that the questions under investigation would determine what research approach would produce useful knowledge for the research (Sekaran & Bougie, 2016).

This research takes a pragmatic approach. Pragmatism favours no specific approach to research. The pragmatic viewpoint allows whatever research methods serve the purpose of the research, including the combination of both positivism and interpretivism (Petersen & Gencel, 2013). Pragmatism views current truth as tentative, which may change over time (Sekaran & Bougie, 2016)

The researcher's aim is to understand the use of agile practices in the coordination between software development and IT operations activities. In particular, the study will investigate DevOps implementation based on the practitioners' perceptions. A pragmatic approach is thus suited for this investigation considering the socio-technical nature of software development. Phases 1 and 2 of the research are mostly influenced by interpretive philosophy (grounded theory is used to generate theories). Phases 3 and 4 are more of a multi-method approach, as a model is developed and evaluated based on multiple philosophies.

3.2.2 Research Logic

Research logic refers to the method of reasoning adopted by the research (Machamer & Silberstein, 2008). It is broadly classified as deductive (top-down approach beginning with theories), inductive (bottom-up approach, from specific observations to broader generalisations), or abductive approach (begins with observation and seeks conclusions from them).

This research takes an abductive reasoning approach. The research is an investigation of DevOps implementation software development companies. The DevOps concept and its practices are still ambiguous in the available literature. The researcher consequently believe that beginning with theories or focusing on specific aspects of the concept would limit an in-depth examination of the intended inquiry. The research begins with exploratory phases of observations and identification of patterns, then the formulation of hypotheses and theories based on analysis of findings. Abductive research guides the researcher's effort to develop 'creative interpretations' of the phenomena studied (Charmaz, 2009). The study takes a further step to develop an adaptive model founded on the theories and evaluates the model with a software development team.

3.2.3 Methodological Choice

Three common approaches or methodological choices to conduct research are quantitative, qualitative, and mixed methods. The methods are designed for specific types of research questions (Williams, 2007).

In line with the pragmatic nature of this study, the research questions under consideration determine the method to adopt. A qualitative method is employed to investigate questions related to the interactions between software development teams and IT operations, the complexity of human behaviours, and DevOps implementation in the agile software development industry. The qualitative approach helps the researcher explore, understand, and interpret the underlying motivation, opinion, and possible reasons for observed phenomena (Creswell & Creswell, 1994, 2017). The investigations are carried out by collecting descriptive and conceptual data through case studies and grounded theory study (Williams, 2007). Mixed methods are used to develop and evaluate an adaptive DevOps strategy implementation model.

3.2.4 Research Strategy

Several strategies are employed in conducting research, depending on the type of inquiry. Quantitative research is mostly associated with experiments, surveys, and archival research. Other research strategies like exploratory, case study, ethnography, action, grounded theory, and narrative inquiry are related to qualitative research (Williams, 2007).

This study employs multiple strategies. Essentially, it is a case study research with data analysis informed by grounded theory, aimed at influencing practice. **Case study research** (Merriam, 1988). is concerned with the investigation of empirical data collected from a multifaceted real-life occurrence, over a period. Case study is widely used in software engineering research (Iyawa et al., 2016; Boehm & Huang, 2003). Runeson & Höst (2009) provides guidelines for carrying out case study research in software engineering. A case study has defined boundaries and generates an elaborate description and in-depth understanding of the phenomenon under study. The case study in phase 1 is DevOps implementation from the perspective of software practitioners doing DevOps for at least 2 years. The case study for phase 2 is DevOps implementation from the perspective of software practitioners that have led the movement in organisations.

Grounded theory research is concerned with the construction of theories through the collection and analysis of data, without preconceived hypotheses. Grounded theory begins with a broad query about a topic and provides an in-depth understanding of social behaviours through the application of inductive reasoning (Heath & Cowley, 2004). Grounded theory followed a well-established set of techniques and procedures. Grounded theory enabled the investigation into the multifaceted nature of DevOps-aware software development from practitioners' perspectives, and the generation of theories.

A DevOps implementation model is developed and evaluated with a team of software developers within an organisation, based on prior-generated theories. Following the evaluation, a physical instantiation of the model is created in a GitHub repository to enable practitioners to implement the model and keep track of its usage. This is more related to action research. Action research is a strategy in which the action researcher is also a participant in the research. Action research is aimed at solving organisational problems. Knowledge is co-generated with collaborating participants and the research culminates in transformative change (Clark et al., 2020).

3.2.5 Research Time Horizons

Research time horizons define the time taken to research the phenomenon. Ideally, this should be dependent on how much time is required to gather relevant information (Philips et al., 2008). The time horizon however is dependent on the availability of research sites and resources. Research could choose either cross-sectional or longitudinal studies.

This research employs cross-sectional time horizons. Phases 1 and 2 uses cross-sectional study involving semi-structured interviews to investigate DevOps implementation across multiple practitioner sites. This time horizon is also used in the evaluation of the developed model. **Cross-sectional study** involves the collection of data from different samples, over a period, to study a phenomenon (Olsen & St George, 2004). Participants are selected based on specific variables of interest. A cross-sectional study allows the researcher to investigate numerous characteristics and their relationships. Cross-sectional studies do not culminate in cause-and-effect conclusions of variables.

3.3 Research Purpose and Research Outcome

Research purpose defines what the research aims to achieve (Robson & McCartan, 2015). The four purposes of research include exploration, description, explanation, and prediction. To answer the question of what software engineering research is for, Beecham et al. (2013) conducted a study to ascertain the impact of Global Software research on practice.

The eventual purpose of this research is that its findings will be used in some way to influence the practice of DevOps and its implementation in organisations. This aligns with Beecham et al. (2013)'s conclusion that studies such as this are performed largely to solve industry-related problems. The repository created in this study will also contribute to future research on the subject matter.

The research outcome is the result and endpoint of the research. The research outcome is categorised into basic research and applied research(Robson & McCartan, 2016).

This research begins with the exploratory aim of gaining a better understanding of DevOps and its implementation in software development organisations. Exploratory research is an inquiry into research questions with little or no previous investigation, or a problem that lacks clear definitions. Exploratory research is mostly a preliminary examination of a broad idea that narrows to specific problems. It involves extensive data collection for an in-depth study of the phenomenon. Findings from exploratory research serve as a focus for future research. Building on initial findings, the study further explores the factors influencing organisations choices of strategies for DevOps implementation. Thereafter, a DevOps strategy implementation model is developed and evaluated based on findings.

Findings from applied research are usually applicable and may be implemented to make improvements or changes to an existing situation(Robson & McCartan, 2016). As an outcome, this is applied research as it tries to solve real-world problems by applying the developed model to improve software development practices within a development team in an organisation. The classification of this study is summarised in figure 3.2

3.4 Research Approach

The research design refers to structuring the research in such a way that the different components of the investigation can integrate to provide logical answers to the research questions (Maxwell, 2012). This research is divided into four (4) phases.

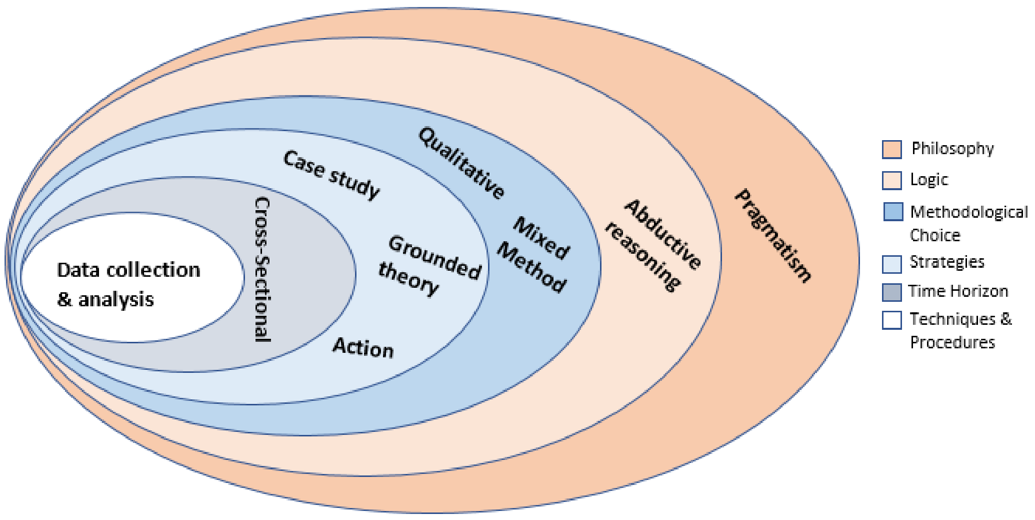


Figure 3.2: Classification of the research

A brief description of the phases follows immediately and is summarised in Figure 3.3. Section 3.5 to 3.7 provide details of the activities and strategies employed in each phase.

Phase 1

Phase 1 is exploratory. It involves data collection on DevOps in general, through semi-structured interviews of software development practitioners. Data analysis is informed by grounded theory.

Phase 2

Phase 2 builds on the findings from phase 1. It involves a more defined data collection on DevOps implementation in organisations. This phase is explanatory. The primary source of data is through semi-structured interviews of software practitioners directly involved in the implementation of DevOps in their organisation. Data analysis is also informed by grounded theory.

Phase 3

Phase 3 is the development of a DevOps strategy implementation model. This model is based on findings from phases 1 and 2, and existing literature.

Phase 4 The evaluation of the model is carried out in this phase. Arising from the evaluation, an instantiation of the model is created in a GitHub repository.

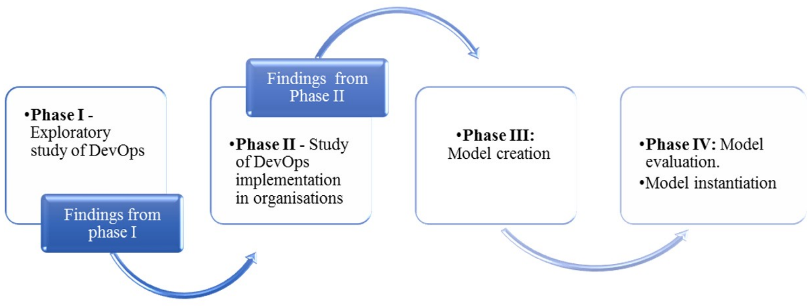

•Phase I - Exploratory study of DevOps
Findings from phase I
Findings from Phase II
•Phase II - Study of DevOps implementation in organisations
•Phase III: Model creation
•Phase IV: Model evaluation.
•Model instantiation

Figure 3.3: Summary of Research Phases

3.5 Phase 1 - Exploring DevOps Implementation in Practice

Phase 1 of the study is an investigation into the understanding and implementation of DevOps in practice. This phase is informed by Grounded Theory approach basically due to the ambiguity of the subject of DevOps in literature and the scarcity of empirical research into its implementation in practice. Following the guidelines of Grounded Theory will allow for the analysis of new concepts from the data collected and the generation of theories grounded in the data.

Grounded theory is considered a suitable qualitative research method to study the social interactions characteristic of software development. The method has been used in numerous Information systems research, especially to study agile software implementation (Hoda et al., 2012; Len et al., 2015; Patton, 2002). Grounded theory is thus suited for my investigation into the implementation of DevOps in software development and software-intensive organisations.

For a basic understanding of the concept, I engaged in light literature review as prescribed in classical Grounded Theory (Glaser, 1978; Hoda et al., 2010). This facilitated effective discussions during interviews. However, an in-depth study was done to understand and carry out Grounded Theory.

The research sites, data collection, and data analysis of phase 1 is described in sections 3.5.1 to 3.5.3.

3.5.1 Research Sites

This section provides short descriptions of some sites of the research. Organisations represented in the study are SMEs and large businesses (intensive software development companies, financial and public institutions) based in the UK, Netherlands, and Africa. The diversity in the research sites provides richness to the data and lends credence to the results. One of the participating organisations is a multinational bank in the Netherlands that deliver services to corporations and other financial institutions through in-house solutions deployed both on-premises and cloud-based platforms. The company is one of the largest in the world and has a significant presence in Europe, with offices around the world. Another is a large international software development company based in Africa. Some of the organisations have collocated teams, while others were geographically distributed. Table 3.1 summarises the demographics of the participating organisations.

As the primary unit of analysis is software development practitioners doing DevOps for at least 2 years, a detailed description of the study participants is presented in Appendix F, rather than that of the organisations. Eleven practitioners from nine organisations participated in the study.

DBA1 and DBA2 were both interviewed and have thus been included in the list of participants. However, the description of their processes was not particularly DevOps related. The data gathered from these participants were excluded from the data analysis. Table 3.2 summarises Phase 1 participants' descriptions.

Table 3.1: Description of organisations in Phase I of the study

Organisation	FinCo1	FinCo2	FinCo3	FinCo4	ITCo	RegCo	FreeCo	PubCo	FinCo5
Size	Large	SME	SME	Large	Large	SME	SME	Large	SME
Business Type	Financial	Financial	Insurance	Financial	IT Consulting	Regulatory	IT Consulting	Public	Financial
Team Location	Distributed	Co-located	Co-located	Distributed	Distributed	Co-located	Co-located	Co-located	Co-located
Team Types	Developers DevOps Ops	Developers DevOps Ops	Developers DevOps Ops	Developers DevOps Ops	Developers DevOps	Developers DevOps	Developers Ops	Developers Ops	Developers
Tools	GitHub, Kubernetes, Ansible, Docker, Azure DevOps, Terraform, Istio	GitHub, SonarCube, Docker, Azure DevOps, Selenium, Veracode, Slack	Kubernetes, GitLab, Ansible, Docker, Bamboo, Jenkin, Terraform, Vault	Jenkins, Jira, Slack, Gitlab	Github, AWS cloud formation and other AWS tools, New Relic, Slack, Terraform	Gitlab, Slack, Kibana, Grafana, Jenkins, Jira, Terraform	Azure DevOps, Skype, Slack, Github	-	Github, Docker, Ansible, Azure DevOps, Terraform, Slack
Practices	Scrum meetings CI/CD	Scrum meetings CI/CD	Scrum meetings CI/CD	Scrum meetings CI/CD	Scrum meetings CI/CD	Scrum meetings CI/CD	Scrum meetings CI/CD	Scrum meetings	Scrum meetings CI/CD
Software Methodology	Scrum, Spotify	Scrum	Scrum, Kanban	Scrum, Spotify	Scrum	Scrum, Kanban	Scrum	Scrum	Scrum

3.5.2 Data collection

This section describes the data collection process. The source of data collection was interviews with 11 practitioners across nine organisations (as shown in Table 3-1). The interviews were conducted over four months in 2019.

3.5.2.1 Recruiting Participants

The data collection process begins with the technique of initial purposive sampling, due to the difficulty in getting organizations to participate in research. Purposive sampling is a non-random sampling technique where the researcher intentionally

Table 3.2: Description of Participants - Phase 1

Participant Code	Job Title	Years of Experience in SD	Years of DevOps practice
DvOps1	DevOps Engineer	18	4
DvOps2	DevOps Engineer	12	7
DvOps3	DevOps Engineer	20	4
DvOps4	Senior DevOps Engineer	7	3
DvOps5	Network DevOps Engineer	31	5
DvOps6	DevOps Engineer	9	3
Dvr1	System Developer	14	3
Dvr2	Software Engineer	3	3
Dvr3	Software Developer	15	2
DBA1	Oracle Apex Developer	9	-
DBA2	Oracle Apex Developer	19	-

chooses participants based on specific qualities they possess(Etikan et al., 2016). Request for participation was sent to suitable professional contacts who themselves participated and further made recommendations to other practitioners. The participants were from different functional areas of software development. This includes developers, IT operations professionals, DevOps engineers, etc.

Prior to the recruitment of participants, Ethics approval to carry out the study was obtained from the University of Salford with approval number STR1819 – 51 (see Appendix A). Each participant was given an information sheet that explained the research and told them that the interviews will be recorded. The consent form included options for their choice of anonymity. Interviews were conducted over Skype and lasted an average of 45 minutes. A bespoke semi-structured interview guide was designed, which also contained a range of open-ended questions relating to practitioners' perceptions and practices of DevOps (see Appendix B). The questions in the interview guide were divided into sections of organisational structure, agile roles, developers and operations coordination, software development process, and code deployment. Initial questions were generated from both experience with investigating agile methods and a light literature review of DevOps. These questions passed through several iterations of reviews by the researcher and were modified and evolved as data collection progressed following a constant comparison process.

During the interviews, questions were tailored to suit the interviewee's role. Questions in sections that did not apply to the role of the interviewee were not asked. The open-ended questions were asked at the end of the semi-structured sections. These open-ended questions provided an opportunity for the participant to discuss topics in the interview guide further, or raise other related issues not mentioned (Adams, 2015). All interviews were recorded with the consent of the participants. Participants' consent were mostly collected on a form (see Appendix C). The form contained information about the study and a request for consent with options. This form was part of the documents for Ethics approval. Recording the interview helped the interviewer concentrate on the topics of discussion rather than on note-taking.

3.5.2.2 Data Transcription

Recordings of the interviews were transcribed verbatim to avoid distortion in meaning (Appendix E). The researcher also tried to capture non-verbal communication during the interviews such as laughter and pauses by manually transcribing the interviews. These non-verbal communications enrich the context of what is being said by the participant. All interviews were conducted in the English language. This eliminates the loss of words through translation. Transcripts were transferred to Nvivo, a qualitative analysis software.

3.5.3 Data Analysis: Based on Grounded Theory

Data analysis of phase 1 follows the prescription of classical grounded theory. In line with the guidelines of the method, phase 1 involves four main aspects of data analysis: open coding, memoing, constant comparison, and saturation.

3.5.3.1 Open Coding

I began with the identification of concepts found within the interview transcripts (Glaser & Strauss, 1967), which involved line-by-line coding of participants' responses without any pre-determined codes. The authors used brief descriptive phrases to represent codes e.g. communication, DevOps enablers, DevOps teams' responsibilities, etc. In the first instance, the codes were handwritten onto hard copies of the interview transcripts, producing 21 codes. A second transcript was independently coded using the Nvivo software, from which 28 codes emerged. In Nvivo, the coding process involved highlighting selected text and creating a node to describe the selected text. Figure 3.4 shows the formation of codes from interview quotes.

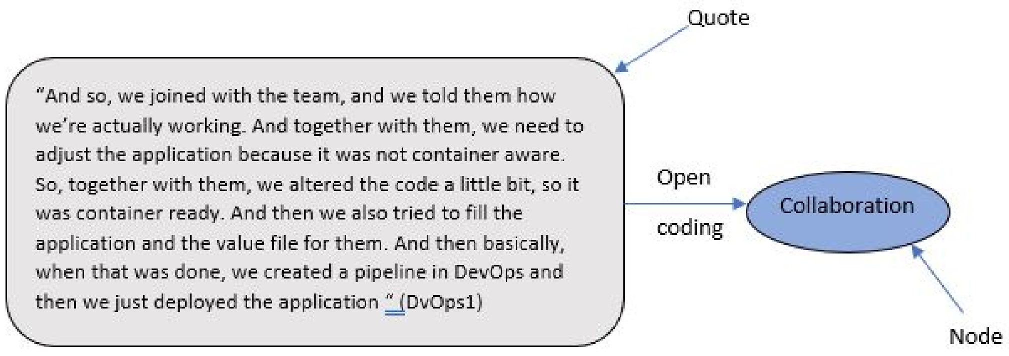

Quote
"And so, we joined with the team, and we told them how we're actually working. And together with them, we need to adjust the application because it was not container aware. So, together with them, we altered the code a little bit, so it was container ready. And then we also tried to fill the application and the value file for them. And then basically, when that was done, we created a pipeline in DevOps and then we just deployed the application " (DvOps1)
Open coding
Collaboration
Node

Figure 3.4: Open coding process

After an initial comparison of the two independent transcripts, the codes were merged into a single set of codes. These were preliminary codes that evolved as data analysis progresses. Subsequent transcripts are coded with already identified codes as well as newly identified codes. Data was then grouped into categories using concept classification (Glaser, 1978), which becomes saturated as new data is (Glaser & Strauss, 1967). The data analysis steps achieved at this stage are shown in Figure 3.5.

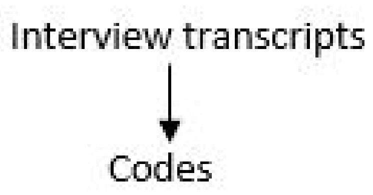


Figure 3.5: Data analysis steps

3.5.3.2 Constant Comparison Method

I used a constant comparison technique to iterate between data collection and analysis, constantly comparing data within itself and other instances of the same case, without any preconceived outline. The technique was used to refine categories and their properties, define and write the theory (Glaser & Strauss, 1967)(Adolph et al., 2011). This approach is "close to the common-sense approach which one might use when trying to understand something which is complex and puzzling" (Robson & McCartan, 2015). Figure 3.6 is an example of a "within instance" comparison and merging of codes.

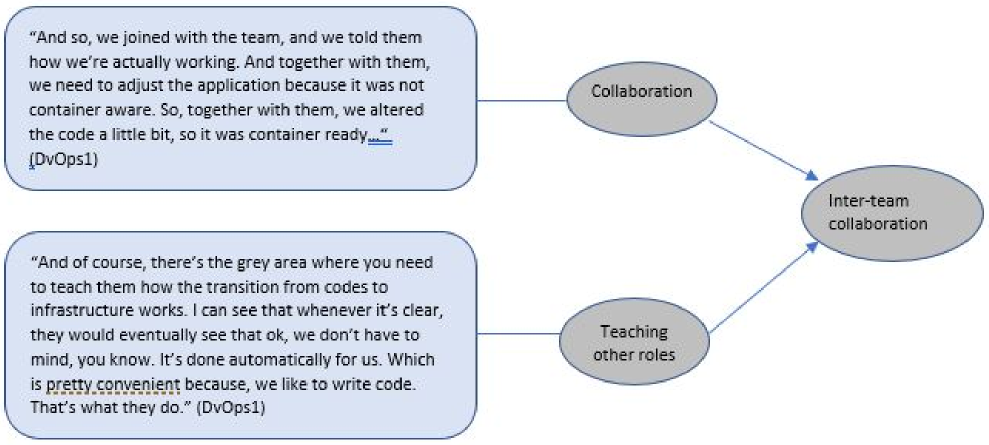

"And so, we joined with the team, and we told them how we're actually working. And together with them, we need to adjust the application because it was not container aware. So, together with them, we altered the code a little bit, so it was container ready..." (DvOps1)
Collaboration
Inter-team collaboration
"And of course, there's the grey area where you need to teach them how the transition from codes to infrastructure works. I can see that whenever it's clear, they would eventually see that ok, we don't have to mind, you know. It's done automatically for us. Which is pretty convenient because, we like to write code. That's what they do." (DvOps1)
Teaching other roles

Figure 3.6: Constant Comparison process

The data analysis steps achieved at this stage are shown in Figure 3.7.

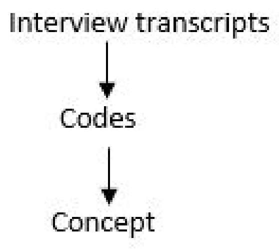


Figure 3.7: Data analysis steps

3.5.3.3 Memoing and Sorting

In this research, I used memos to capture and refine concepts, and to express the relationship between concepts identified using open coding as they develop into categories (Glaser, 1978). Based on the identified codes, brief notes on topics were made containing quotations from transcripts as primary evidence, from which 10 memos emerged. Memo writing helped to elucidate and converge ideas originating from the codes, and sharpen categories evolving as new transcript data is added (Glaser & Strauss, 1967). This is a key stage in theory generation (Adolph et al., 2011). Figure 3.8 is an example of the emergence of memos from codes and categories generated from the data.
The memos were evaluated to identify relationships which, combined with swim lane diagrams of functional areas of software development, formed the basis for my initial findings. The memos were constantly updated throughout the data analysis process.

As the data collection seemed near saturation, the memos were sorted to form a theoretical outline. Sorting “puts the fractured data back together” (Glaser, 1978). The sorting was based on the relationship between categories, comparing one memo to the other. This formed the initial structure of the theories. The data analysis steps achieved at this stage are shown in Figure 3.9.

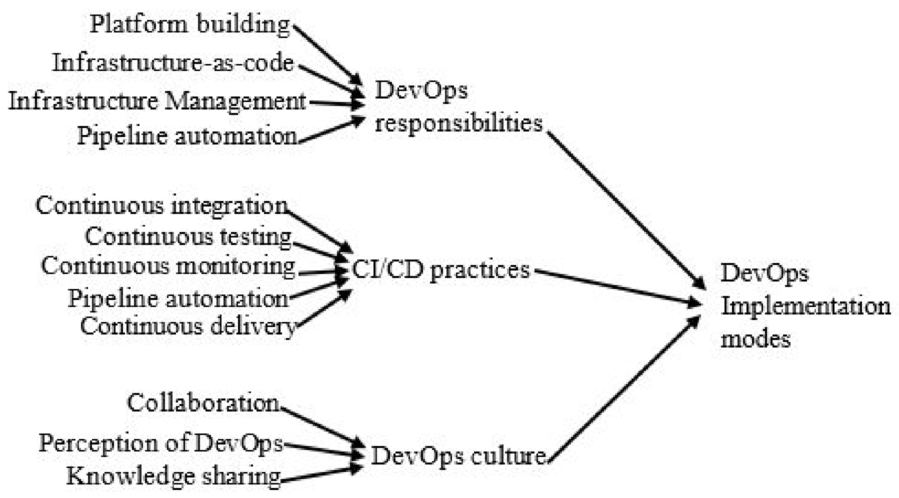


Figure 3.8: Memo Generation process

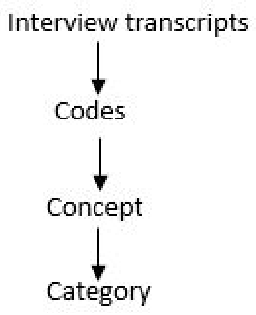


Figure 3.9: Data analysis steps

3.5.3.4 Saturation

As expected, evidence began to converge at the later stage of the research, and the addition of new interviews had less and less impact on the categorisation. Saturation is said to have occurred in the research when new categories no longer emerge. There are arguments for theoretical and data saturation, and some hybrid forms. While the aim of the study is not to distinguish between theoretical and data saturation, it adopts Glaser (1978) definition of “Saturation means that no additional data are being found." Emerging theories were identified from memos of transcripts, more data is collected and coded, and constantly compared with the existing codes and memos until no additional information seemed apparent. Although this is not a claim of saturation, I believe that at this point in my work, I can examine the theory emerging from the analysis of the data.

The data analysis steps achieved at this stage are shown in Figure 3.10.

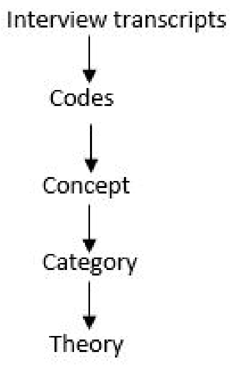


Figure 3.10: Data analysis steps

Figure 3.11 summarises the research method employed in phase 1 of the study.

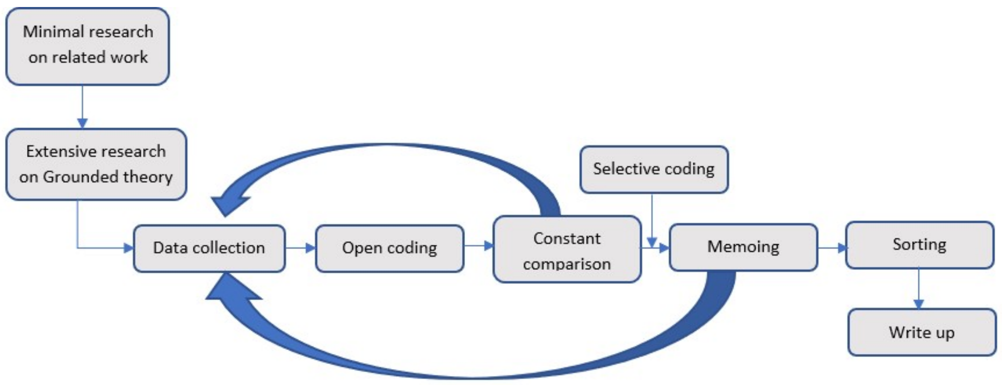


Figure 3.11: Phase 1 research methodology

3.6 Phase 2 - DevOps Implementation strategies and Elements of influence

Phase 2 aims to investigate the rationale and evidence-based decision-making in organisational DevOps implementation, and the elements of influence. This phase is also guided by grounded theory methods. This is because this aspect of the study is also exploratory, although it builds on findings from the previous phase. I describe the research sites, data collection, and data analysis in the sections following.

3.6.1 Research Sites

I provide a brief description of some participating organisations in this section. Organisations represented in the study are all large enterprises (software development companies as well as software-intensive companies, in the financial, healthcare, and industrial sectors based in the UK, USA, and Africa). For instance, one of the participating organisations is an international airline based in the USA. They deliver services to millions of customers through in-house solutions deployed on both on-premises and cloud-based platforms. Another is a large international software development company based in Africa. Some of the organisations had co-located teams, while others were geographically distributed. Classification of organisations in this study is based on staff count, adopted from the EU Recommendation 2003/361.

The primary unit of analysis for this phase was software development practitioners and senior IT managers that have led DevOps implementation in organisations. This was to provide us with an organisational level perspective of the elements that influence DevOps implementation. Seventeen practitioners each from a separate organisation participated in this phase. Appendix F describes the participants.

Table 3.3 summarises Phase 2 participants' descriptions.

Table 3.3: Participants' description

Code	Position	SD Experience (years)	DevOps practice (years)
DevCo1_CTO	CTO	31	9
FinCo7_Lead	Senior DevOps Engineer	13	8
MultCo1_Mgr	Engineering Manager	11	5
DevCo2_MD	Principal Consultant	31	9
DevCo3_Lead	Principal Architect	16	6
DevCo4_CEO	Principal Consultant	32	11
DevCo5_Lead	Lead DevOps Engineer	19	8
FinCo8_TM	Technical Manager	29	9
FinCo9_Lead	Global DevOps Lead	16	8
DevCo6_DOps	Senior DevOps Engineer	14	5
DevCo7_Lead	Lead Site Reliability Engineer	11	4
DevCo8_Chief	Chief Consultant	9	9
DevCo9_Chief	Chief Consultant	23	7
DevCo10_Head	Head of Software Dev.	21	7

3.6.2 Data collection - Theoretical Sampling

Theoretical sampling is the process of data collection for generating theory whereby the analyst ”jointly collects, codes, analyses the data and decides what data to collect next and where to find them, to develop his theory as it emerges.” (Glaser, 1978)

3.6.2.1 Recruiting Participants

In the second phase, data collection was through interviews with 14 senior IT managers and DevOps transformation leads each from various organisations. Some participants also provided us with documentary evidence of their processes. The interview was conducted over seven months in 2020, following the process previously described.

The recruitment of participants for the phase was matched with the primary unit of analysis. This was initially done by a LinkedIn profile search of experts in DevOps groups who claimed to have experience in organisational DevOps implementation. Thereafter, I progressed to recruit participants through the snowball sampling method.

3.6.2.2 Data Transcription

As described in phase 1, interviews were transcribed verbatim and transferred to Nvivo software for analysis.

3.6.3 Data Analysis: Based on Grounded Theory

Data analysis of phase 2 also followed the techniques of open coding, constant comparison, memoing, theoretical coding, and saturation

3.6.3.1 Open Coding

Open coding followed the same process described in phase 1: line-by-line coding of transcripts using short descriptive phrases. Seventy-nine (79) codes were generated from open coding in phase 2. Some codes were merged using a constant comparison process, yielding 40(forty) core codes.

3.6.3.2 Constant Comparison Method

Constant comparison in this phase involved within the transcript, between transcripts of the same phase, and with codes previously generated in phase 1, to determine relationships. It follows the same process previously described.

3.6.3.3 Memoing and Sorting

Memoing in this phase followed the same process described in Phase 1. Examples of the memo generation process are shown in Figure 3.12 - Figure 3.14.

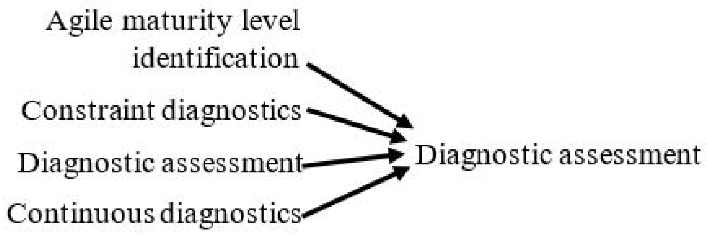


Figure 3.12: Diagnostic assessment memo generation

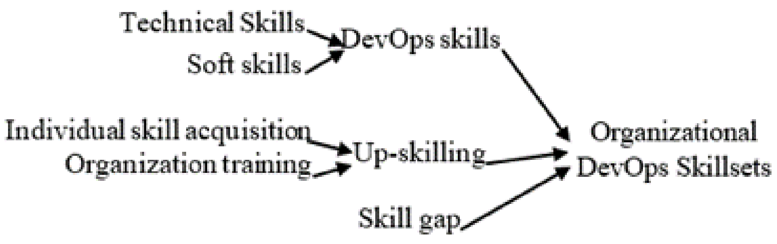


Figure 3.13: Organisational DevOps Strategies memo generation

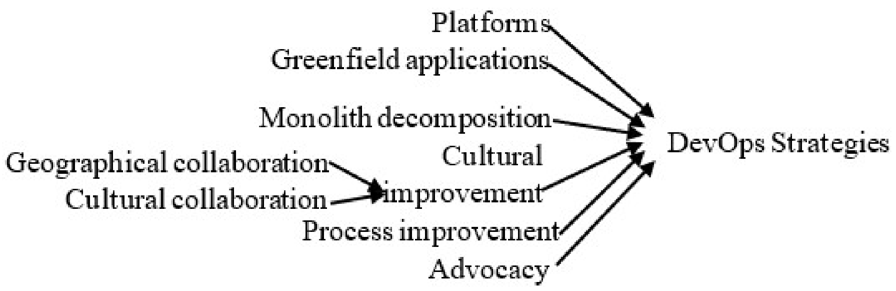


Figure 3.14: DevOps strategies memo generation

3.7 Phase 3

3.7.1 Model Creation Process

The model creation is based on theories generated from findings in phases 1 and 2, and literature. This process mainly involved an examination of the models used in tailoring agile software development methodologies and the selection of a suitable model. The selected model was extended to accommodate the findings of the research. These steps are shown in Figure 3.15.

The model consists of 4 parts. Parts 1 and 2 directly relate to the selected tailoring model for agile software methodologies. A comparison between both models is presented in chapter 6. Parts 3 and 4 mainly reflect findings from this study and seek to address the requirements for the implementation of DevOps in an organisation.

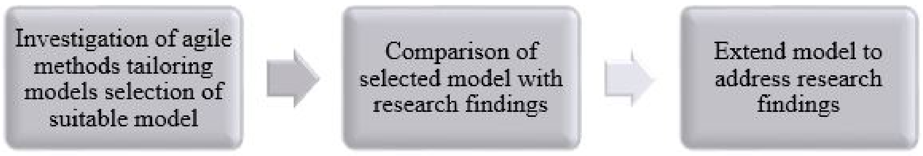

Investigation of agile methods tailoring models selection of suitable model
Comparison of selected model with research findings
Extend model to address research findings

Figure 3.15: Model creation process

3.8 Model Evaluation

3.8.1 Evaluation Site

The evaluation of the created model was carried out with DevCo9. DevCo9 is a UK-based international software provider, specialised in oil and gas engineering and management solutions. This includes software such as maintenance and optimisation, safety, risk assessment, as well as inventory and asset management. DevCo9 has a global team distributed around Europe, America, the Middle East, Asia, and Africa. They deliver services to millions of customers through solutions deployed on cloud-based platforms. In-house services are both cloud native and on-premises. The organisation also provides consultancy for studies such as drilling and well operations, assets and integrity management, etc.

3.8.2 Model Evaluation Structure

The evaluation of the model consists of four main parts:

1. A pre-workshop engagement with DevCo9
2. A focus group workshop involving the presentation of the model
3. Recommendation of actions to improve agility in the the organisation, based on the model.
4. The creation of a repository to aid practical usability of the model, based on participants' suggestion.

3.9 Pre-Workshop Engagement with DevCo9

The pre-workshop engagement with DevCo9 was a discovery meeting to evaluate how relevant SIDSEM would be to their context. One of the techniques used for this activity was an interview. A pre-determined interview guide was used which contained questions relating to the software development process and collaboration. Two expert stakeholders were interviewed. Each interview lasted about 40 minutes. The interviews were transcribed and analysed. The discovery meetings also involved a discussion of my research findings and a general description of the model. A participant information sheet (Appendix D) was provided to each participant. As an outcome, a 2-hour workshop was agreed on with a team of software development experts in DevCo9.

A summary of the evaluation activities is presented in Figure 3.16

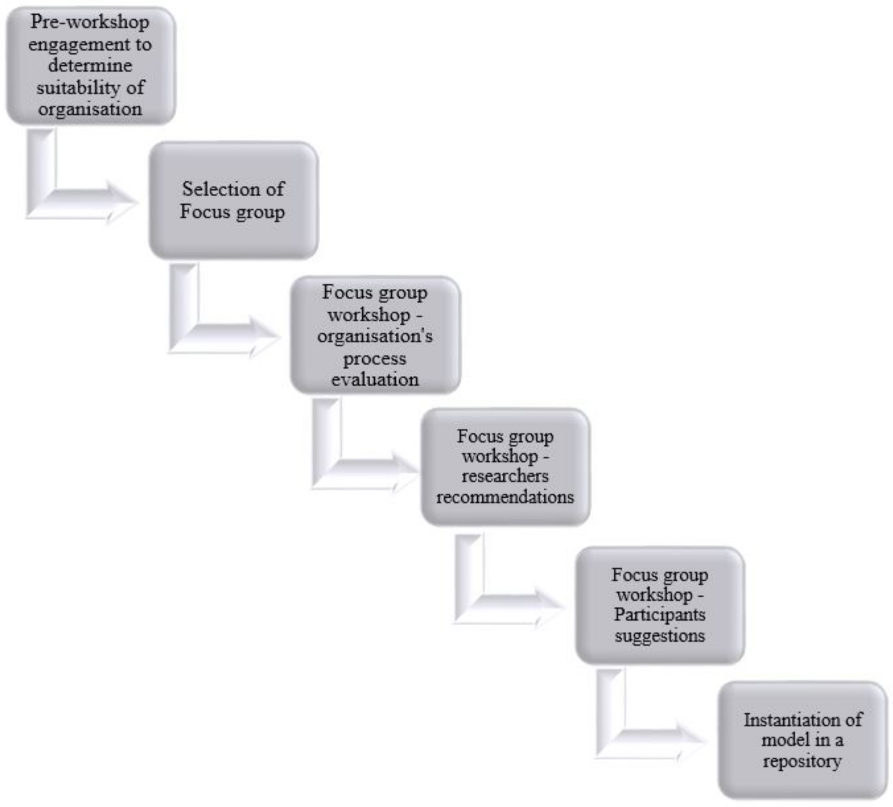

Pre-workshop engagement to determine suitability of organisation
Selection of Focus group
Focus group workshop - organisation's process evaluation
Focus group workshop - researchers recommendations
Focus group workshop - Participants suggestions
Instantiation of model in a repository

Figure 3.16: Model evaluation process

Chapter 4

Taxonomy of DevOps Implementation in Practice, Benefits, and Challenges

4.1 Introduction

This chapter represents findings from Phase 1 of the study. The findings, which can be described as preliminary, were gained from an analysis of interview transcripts. The chapter gives the reader an in-depth exposition of DevOps implementation in practice, as understood by the researcher. I also present a novel taxonomy of DevOps implementation derived from the findings.

The next session describes DevOps in practice. This is followed by approaches to DevOps implementation, then I explore the DevOps teams' responsibilities. Thereafter, I present the identified difference between developers, DevOps, and IT Ops teams. Findings on CI/CD practices, collaboration, benefits, and challenges of DevOps implementation are then presented. Out of the nine organisations that participated in the study, eight had adopted DevOps culture and implemented its practices across the organisation. The remaining organisation has thus been excluded from the finding herein presented.

4.2 Description of DevOps

This section presents the description of DevOps from practitioners' perspectives. The participants generally describe DevOps as improved collaboration between developers and IT Ops teams. Some interviewees also described DevOps as end-to-end automation of the software development pipeline, aimed at providing better software quality, and creating a seamless workflow of products in the shortest possible time. These descriptions appear to fall into two main groups: DevOps as a culture, and as a job description.

4.2.1 DevOps as a Culture

Based on participants' descriptions, the researcher understands "DevOps as a culture" to mean a way of thinking and approach to working. This "culture" is portrayed as an intentional collaboration between developers and IT Ops specialists. DevOps as a culture was widely reported among interviewers. In FinCo1, the DevOps team assists developers in re-configuring their codes for containerisation. "And, so we joined with the team and we told them how we're actually working. And together with them, we tried to, we need to adjust the application because it was not container aware. So, together with them, we altered the code a little bit, so it was container-ready". [Finco1_DOps1]. Knowledge is shared and there is a mutual understanding of basic activities across the boundaries of teams. This informal knowledge sharing is said to be achieved through the collaborative resolution of code-related challenges. Similarly, [Finco1_DOps2] mentioned knowledge flow from developers to DevOps Engineers: "So as I mentioned, there's a couple of proper developers, we have a great resource because they teach us, nope, sit back, look. What are you trying to achieve? And write it properly from scratch rather than just couple together something".

DevOps teams in these organisations are conversant with the codes of developers and help where necessary. Developers are also made aware of how the automated infrastructure works, though not directly involved in its creation or maintenance. According to some practitioners, a level of confidence is brought about by a basic understanding of other aspects of the process and familiarity with the other actors. Intra-team collaboration is reported as brainstorming and coding together when issues are encountered. Collaboration in FinCo2 involves the DevOps team creating users' stories from requirements, breaking them into manageable tasks, and delegating these tasks to developers through Azure DevOps.

4.2.2 DevOps as a Job Description

Some interviewees had the job title of 'DevOps Engineer' and worked in distinct DevOps teams or departments. "We don't actually have developers in our team. So, in our case... it's just DevOps" [Finco1_DOps1]. They further described their team as "platform builders" for developers, "who support them and host their applications on our platform". Here, DevOps is being presented as a job description, with DevOps Engineers responsible for carrying out "DevOps functions", which I describe in the DevOps teams' responsibility subsection. Some participants expressed concern that having separate DevOps teams might not be the right way to implement the concept, however, others affirmed that the approach allowed developers to focus on providing value for the business, as they are not encumbered with the extra burden of creating and managing automated CI/CD pipeline for their deployments.

4.3 Approaches to DevOps

The results show that organisations adopt several approaches to implement DevOps. Practitioners try to adopt acclaimed DevOps-related best practices in a seemingly suitable way. However, according to participants, the choice of an approach to DevOps implementation is largely unstructured. This means that the determination of the most suitable DevOps practices does not follow any specific guidelines. From my analysis, I present a taxonomy of DevOps implementation: four dissimilar modes describing developers' interaction with Ops, Outsourced Ops, DevOps, and DevOps Bridge teams. It is important to note that the developers' teams encountered in the study include QA and security experts.

4.3.1 Developers-Ops mode

Developers-Ops mode of DevOps implementation depicts the instance where senior developers performed automated infrastructure management alongside development activities in a hybrid cloud deployment environment. The IT Ops team supports the physical infrastructure and on-premises hosted applications, while developers wrote application codes, deployed their codes through CI/CD pipelines, and managed applications themselves. Here, senior developers were seen as the facilitators of DevOps practices. Figure 4.1 is a pictorial representation of Developers-Ops mode.

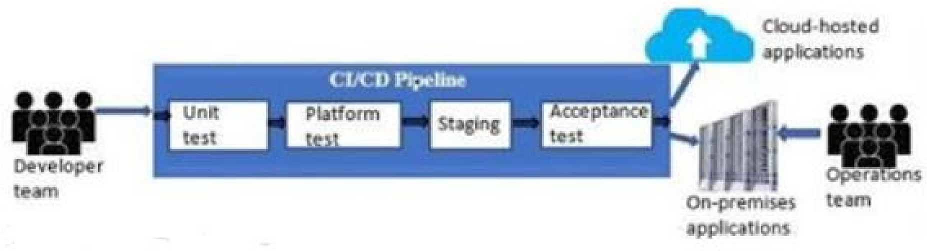


Figure 4.1: Developers-Ops mode

4.3.2 Developers-Outsourced Ops mode

The Developers-Outsourced Ops mode is like the Developers-Ops mode described above. Senior developers also write infrastructure codes to create and manage deployment pipelines, the difference being that its deployment environment is cloud-based, eliminating the need for Ops experts. Figure 4.2 describes the mode.

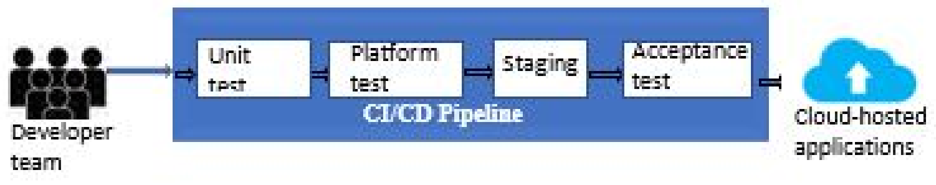


Figure 4.2: Developers-Outsourced Ops mode

4.3.3 Developers-DevOps mode

Figure 4.3 depicts the Developers-DevOps mode where DevOps teams create, deploy, and manage both the cloud infrastructure and deployment pipelines. Developers' applications are also deployed and maintained by the DevOps team. Describing this mode, [Regco_Dvr] said "it allowed developers to focus on providing value for the business". This claim was clearly expressed by some other participants. Here, developers are not responsible for application deployment and management. Completed applications or features are handed over to the DevOps teams for deployment and management, who are the DevOps practices facilitators.

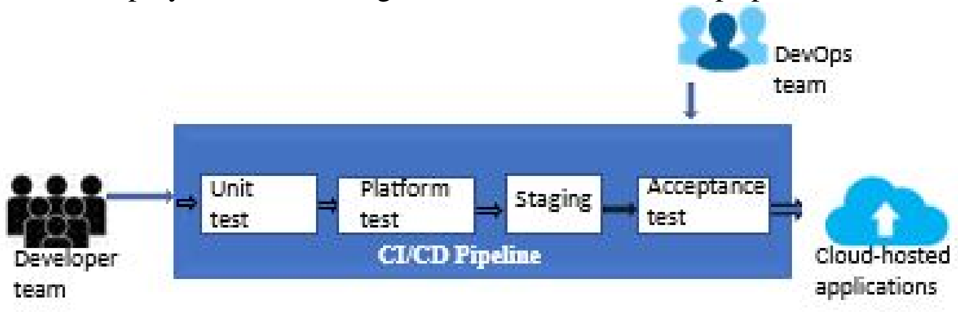


Figure 4.3: Developers-DevOps mode

4.3.4 DevOps bridge team mode

DevOps bridge team mode was the mode widely used in my study. This mode (shown in Fig. 4.4) was found in a hybrid environment of cloud and on-premises deployment. Here, DevOps teams interface with both developers and IT Ops to drive the practices of DevOps like configuration management, continuous integration and continuous delivery, automated testing, deployment, monitoring, and metrics collection. Developers provide business solutions, leaving the creation, deployment, and management of both the cloud infrastructure, virtual systems, and deployment pipelines to the DevOps teams. These teams are provided services of on-premises infrastructure by the Ops team. Essentially, the DevOps teams are customers to the Ops team, and service providers to developers. It is important to note that in this approach to DevOps, everyone is responsible for their actions. Developers create codes, deploy them through CI/CD pipelines, monitor, and manage applications. In the same vein, the DevOps team deploys its infrastructure through the pipelines they create. However, the bridge teams are the facilitators of the DevOps process.

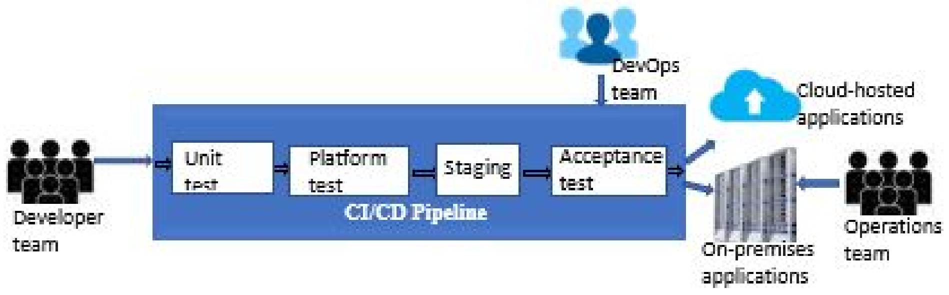


Figure 4.4: DevOps bridge team mode

Further investigation to understand this approach revealed three main classes of activities: provisioning and maintenance of physical systems, function virtualisation and creation of automated pipelines, and development, deployment, and maintenance of applications (Continuous practices).

At the physical level, participants clearly explained that the systems in their on-premises data centers needed to be configured and managed by IT Ops teams. Access is thereafter granted to DevOps engineers, who built automation into these systems. [Finco1_DOps2] describes the situation as "There's one team that does the physical installation, cabling of the hardware. There's another team that does the installation of some sort of the OS, and there's another team that customises that OS and we get access to install our [tools]... So, we are the consumers of it."

Using automation tools, DevOps engineers create pipelines to enable continuous practices such as continuous integration/continuous deployment, continuous testing, etc. Scripts are created for configuration management and the deployment of infrastructure-as-code. In FinCo1, FinCo2, FinCo3, and FinCo4, these activities are solely the responsibility of the DevOps teams. Applications and solutions are subsequently developed and deployed through automated pipelines. The processes are mostly automated, however, code review and user acceptance tests were described as manual processes.

Table 4.1 shows the distribution of the organisations according to their approaches to DevOps implementation. While the aim is not to study the frequency of each approach, it is interesting to point out that the DevOps bridge team approach seems quite popular in the study. The researcher thinks that this might be a result of organisations trying to understand the concept while trying to maintain stability in their operations.

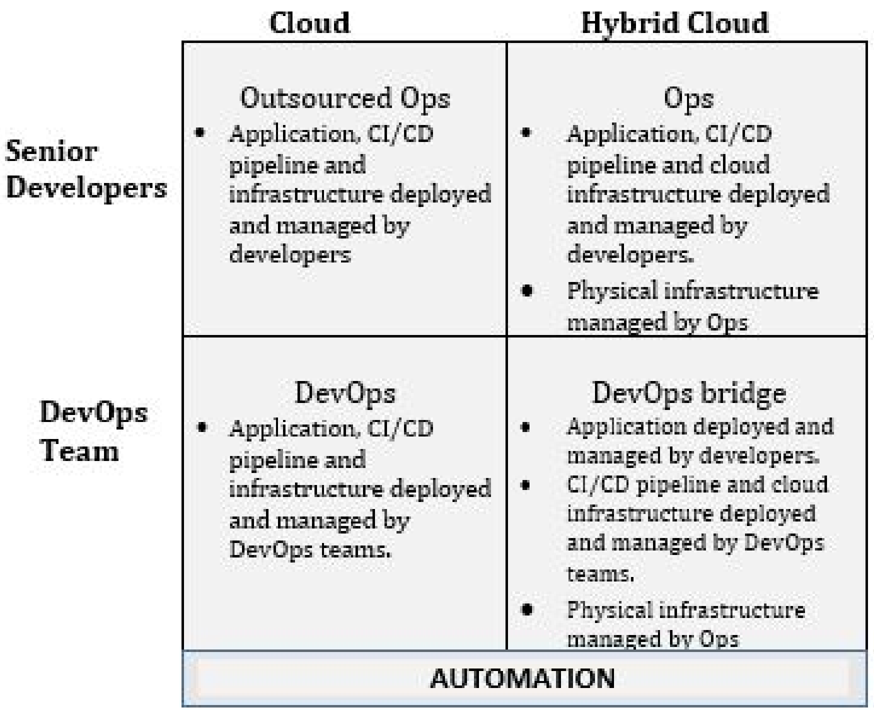
Table 4.1: DevOps implementation approaches in the study

Approach	**Organisation**	**Deployment platform**
Developers-Ops collaboration	FreeCo1	Hybrid Cloud
Developer-Outsourced Ops	FinCo5	Cloud
Developers-DevOps collaboration	ITCo1, RegCo1	Cloud
DevOps bridge teams	FinCo1, FinCo2, FinCo3, FinCo4	Hybrid Cloud

Despite the seeming prevalence of bridge teams in the study, some interviewees thought it was not the right approach to DevOps implementation, as "there is still segregation between development and infrastructure in some way." [Finco1_DOps1]

4.4 Taxonomy of DevOps approaches

Based on practitioners' descriptions already presented, Fig. 4.5 is a quadrant showing a novel taxonomy of DevOps implementation identified in the study. This classification describes the interaction of developers with various teams and presents a summary of the activities in each mode of DevOps implementation. The elements of interest are the application deployment environment, the facilitators of DevOps practices, and the team structures found in their interactions. It might be valuable to study how these elements influence one another in the determination of the approach organisations take to implement DevOps. The taxonomy thus provides a foundation for such investigation, which is beyond the scope of this study. The x-axis in the quadrant shows the application deployment environment. The y-axis shows the facilitators of DevOps practices.



	Cloud	Hybrid Cloud
Senior Developers	Outsourced Ops • Application, CI/CD pipeline and infrastructure deployed and managed by developers	Ops • Application, CI/CD pipeline and cloud infrastructure deployed and managed by developers. • Physical infrastructure managed by Ops
DevOps Team	DevOps • Application, CI/CD pipeline and infrastructure deployed and managed by DevOps teams.	DevOps bridge • Application deployed and managed by developers. • CI/CD pipeline and cloud infrastructure deployed and managed by DevOps teams. • Physical infrastructure managed by Ops
	AUTOMATION	

Figure 4.5: Taxonomy of DevOps approaches

4.5 DevOps Teams' Responsibilities

The DevOps teams in the study are tasked with migration from existing platforms to either cloud-based or an automated on-premises environment, and its subsequent maintenance. Generally, they act as an intermediary between IT operations and developers, providing the means to an end in software development (SD), by creating automated pipelines on both physical and virtual servers to enable continuous integration and continuous delivery. In Finco1 and ITCo1, DevOps teams are organised around specific products. Beyond provisioning automated platforms and maintenance of the environment, however, there are slight variations in the responsibilities of the DevOps teams encountered in the study. For example, the DevOps team in Finco2 is tasked with development cycle automation and tool unification. More than that, they coordinate the activities of the software development process, serving as scrum masters in some instances. As [DvOps2] describes it, "We bring the same tools between developers and operations to like a common ground in the whole of our software life cycle…tracking of the day-to-day activities by organising scrum meetings, of the operations team, the product owners and things like that and the developers to also… aggregating requirements from the business to developers."

The two participants from Finco1 are in two separate DevOps teams, working on cloud and on-premises platforms respectively. Team 1 is a cloud team responsible for onboarding in-house solutions, which are resident in on-premises datacentres, to public cloud platforms. The team creates automated deployment pipelines using Kubernetes, and virtual services to route applications, providing the mechanism for faster deployment. [DvOps1] puts it this way, " I'm part of the cloud team, and this basically is a new team which was launched a couple of months ago. And the goal of the team is to investigate the movement to the public cloud. What we try to do is, we try to give them the right tools. So that basically means that we define the pipeline". Team 2 however works on an on-premises container hosting platform. They create pipelines to enable Continuous integration/continuous delivery. They write codes, peer-review, and deploy pipeline solutions, through automated pipelines.

In RegCo1 and ITCo1, the DevOps team is responsible for all deployments. Developers hand over applications to these teams, who then oversee the journey through the CI/CD pipeline. The team also monitor the applications and function first line of support.

The responsibilities described are shown in the swim lane diagrams in Figures 4.6 to 4.9, based on participants' descriptions of the identified approaches.

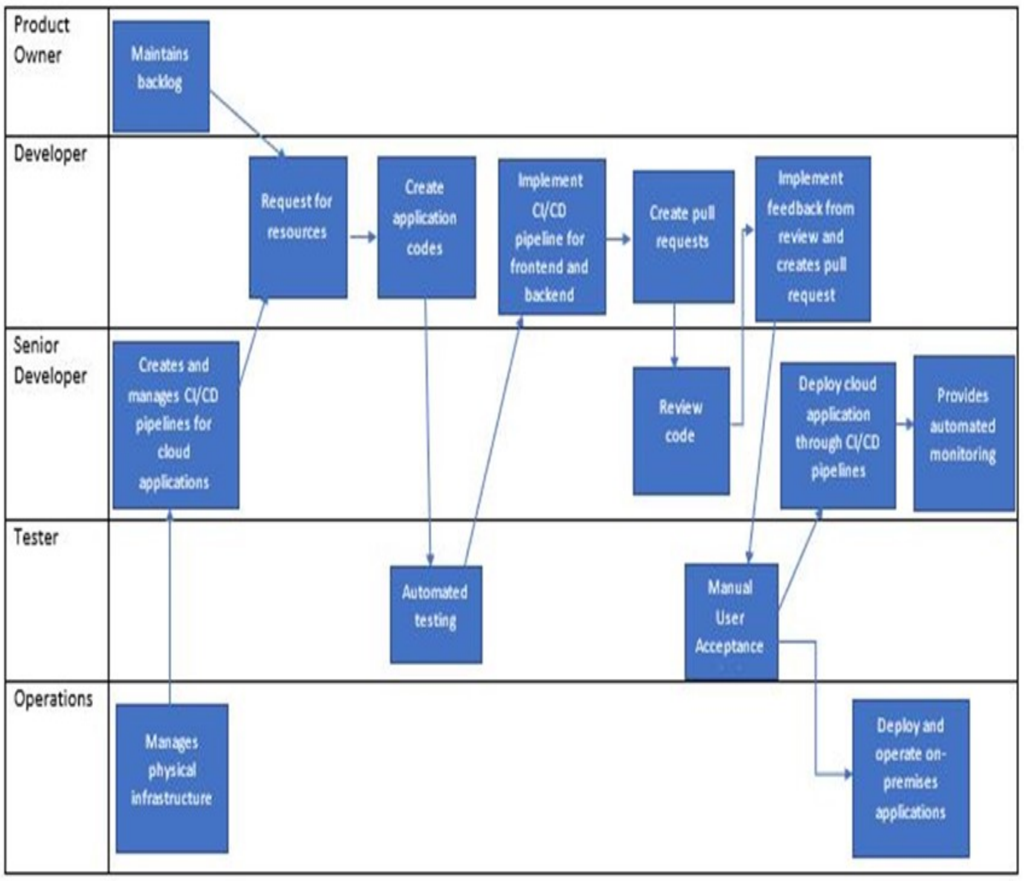

Product Owner
Maintains backlog
Developer
Request for resources
Create application codes
Implement CI/CD pipeline for frontend and backend
Create pull requests
Implement feedback from review and creates pull request
Senior Developer
Creates and manages CI/CD pipelines for cloud applications
Review code
Deploy cloud application through CI/CD pipelines
Provides automated monitoring
Tester
Automated testing
Manual User Acceptance
Operations
Manages physical infrastructure
Deploy and operate on-premises applications

Figure 4.6: Responsibilities in Developers-Ops Mode

4.5.1 Boundaries of Responsibilities

An interesting point observed is participants' insistence on a distinction between "pure developers" and the DevOps engineers and IT operations. For instance, some DevOps engineers interviewed constantly referred to developers as "them".

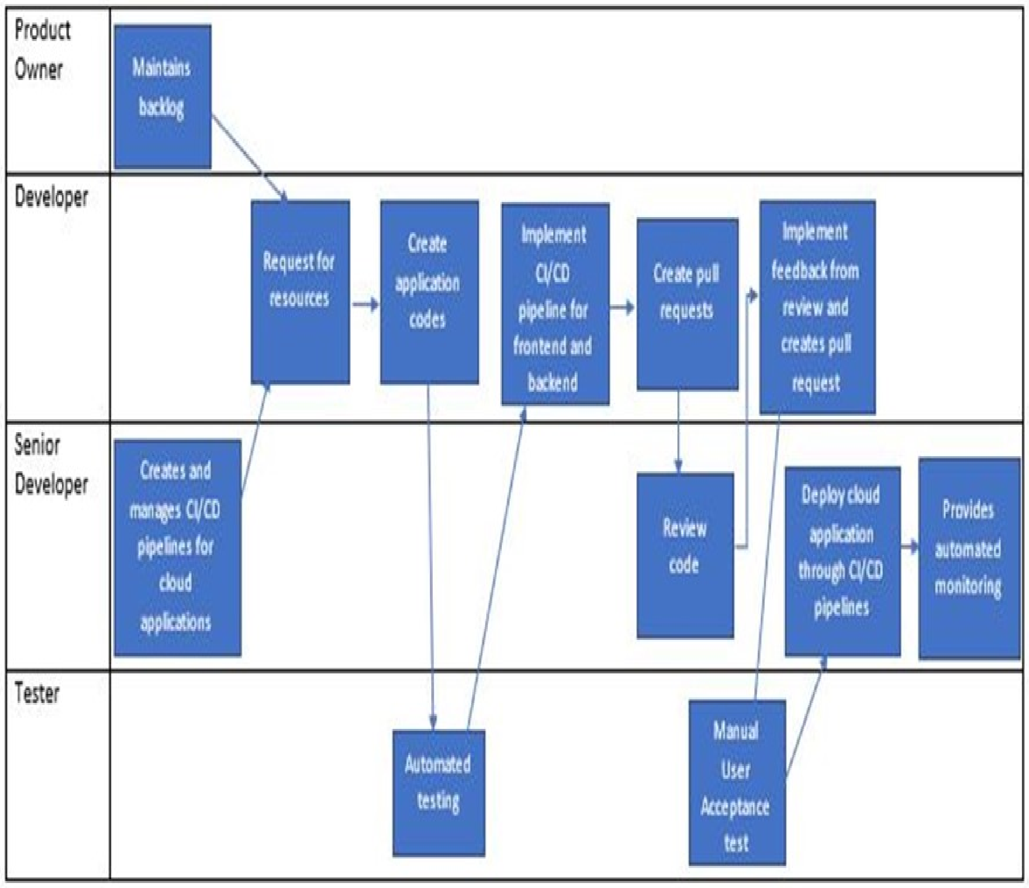

Product Owner
Maintains backlog
Developer
Request for resources
Create application codes
Implement CI/CD pipeline for frontend and backend
Create pull requests
Implement feedback from review and creates pull request
Senior Developer
Creates and manages CI/CD pipelines for cloud applications
Review code
Deploy cloud application through CI/CD pipelines
Provides automated monitoring
Tester
Automated testing
Manual User Acceptance test

Figure 4.7: Responsibilities in Developers-Outsourced Ops Mode

Some participants expressed the view that software developers should spend their time working on their products and not be bothered with what goes on "beneath the hood" of the infrastructure side of the platform if they can deploy solutions seamlessly. As DvOps1 said: "Yes, they're purely development teams. They create APIs, beacon APIs or full-term applications, or just parts of websites...The only thing that we actually want them to do is to write the code and we should know how to start your application. And they create the docker file which comes with it. And we try... from there we try to pick up the rest."

Also, while the DevOps team works to give developers the best tools to get their work done, developers are expected to take responsibility for their products. This suggests boundaries of responsibilities. Another observed instance is the distinction

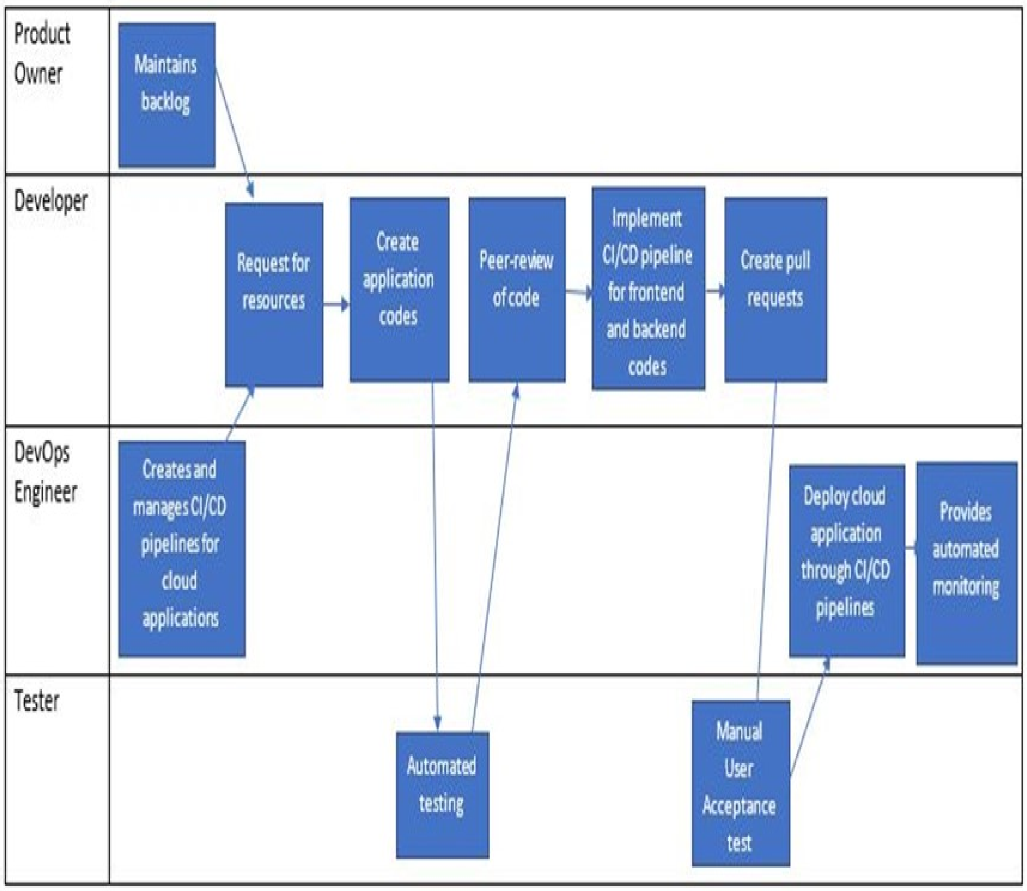

Product Owner
Maintains backlog
Developer
Request for resources
Create application codes
Peer-review of code
Implement CI/CD pipeline for frontend and backend codes
Create pull requests
DevOps Engineer
Creates and manages CI/CD pipelines for cloud applications
Deploy cloud application through CI/CD pipelines
Provides automated monitoring
Tester
Automated testing
Manual User Acceptance test

Figure 4.8: Responsibilities in Developers-DevOps Mode

between DevOps and IT Operations. DvOps3 describes the situation as ”There’s one team that does the physical installation, cabling of the hardware. There’s another team that does the installation of some sort of the OS, and there’s another team that customises that OS and we get access to install our [tools]... So, we are the consumers of it. We don’t really care but we...we’re the next chain in line, we know that there’s many teams before [us]. We expect the other teams to deal with those processes, and deal with their impediments and deliver it” The distinction is seen in the responsibility for product codes and delivery, the deployment pipeline and automated infrastructure management, and the physical infrastructure administration. Table 4-2 summarizes the activities of developers, DevOps, and IT Ops teams identified in the study.

All deployments in the study, from developers and DevOps teams, go through

Table 4.2: Difference between developers, DevOps and IT Ops

Task	**Developers**	**DevOps Teams**	**IT Ops**
Coding	Write code for products and features	Write codes for tools and virtual functions	Write scripts for functions
Continuous Integration	Use CI/CD pipeline to continuously merge code to master branch	Use CI/CD pipeline to continuously merge code to master branch	-
Deployment	Deploys own solutions to both cloud and on-premises platforms using automated tools	Deploys own solutions to both cloud and on-premises platforms using automated tools	Deploys developers' solutions to on-premises physical and virtual systems
Infrastructure management	Manages cloud infrastructure (1 instance)	Automation pipeline management on both cloud and on-premises infrastructure	Physical infrastructure management
Other identified responsibilities (1 instance)	-	Translates requirements to user stories, scrum masters, assigning tasks to developers, project tracking/-monitoring	-

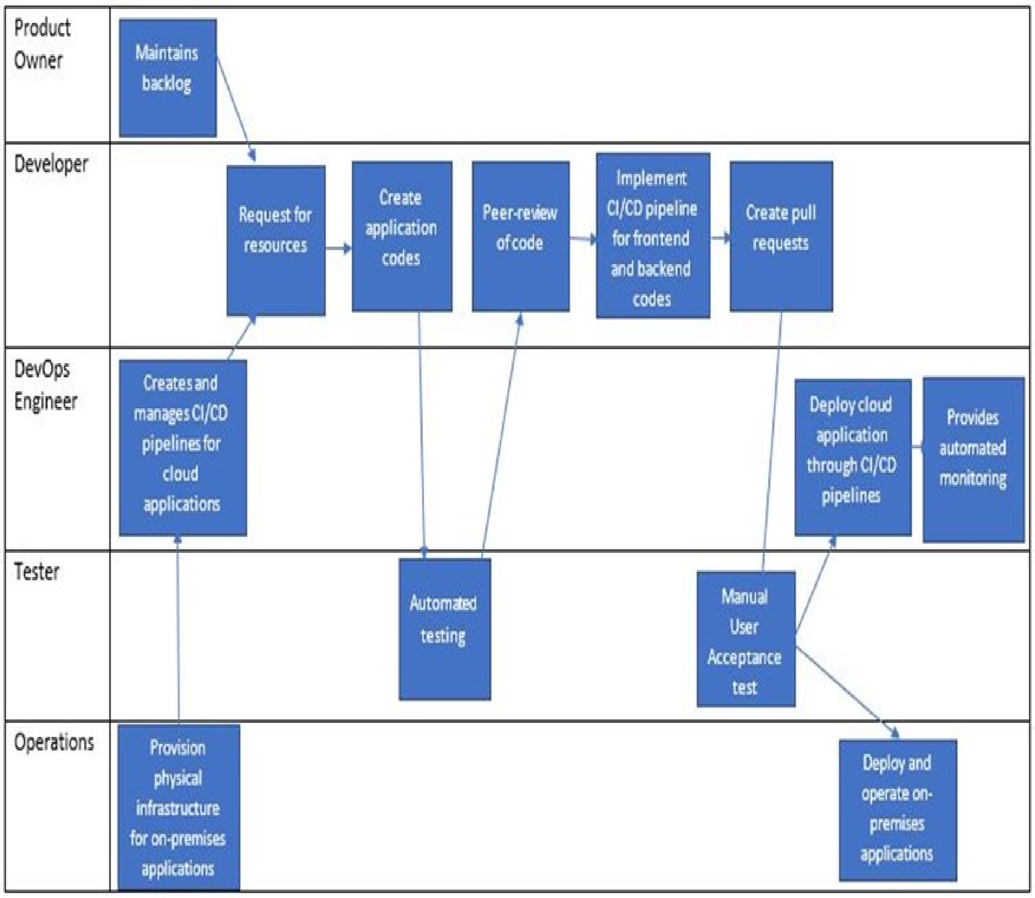

Product Owner
Developer
DevOps Engineer
Tester
Operations
Maintains backlog
Request for resources
Create application codes
Peer-review of code
Implement CI/CD pipeline for frontend and backend codes
Create pull requests
Creates and manages CI/CD pipelines for cloud applications
Deploy cloud application through CI/CD pipelines
Provides automated monitoring
Automated testing
Manual User Acceptance test
Provision physical infrastructure for on-premises applications
Deploy and operate on-premises applications

Figure 4.9: Responsibilities in DevOps Bridge Mode

automated pipelines. Here, I see an intersection of an activity (deployment) between the developers and the DevOps team, however, Table 4-2 creates a distinction between the types of deployments carried out by each group.

4.6 DevOps Practices

4.6.1 Continuous Integration, Continuous Delivery (CI/CD), and Tooling

CI/CD practices are quite similar across the organisations in my study. All the participants described a push CI/CD pipeline model. Developers commit codes to the repository via an automated pipeline. Following predefined rules, the code is built. Depending on the build outcome, a successful build is moved through test environments and is finally deployed to production. Deployment is triggered by a manual approval process after verification of acceptance criteria in the test environment. This model of deployment is also practised by the DevOps teams. Their infrastructure scripts go through the CI/CD pipeline to production. Figure 4.10 is a representation of the pipeline.

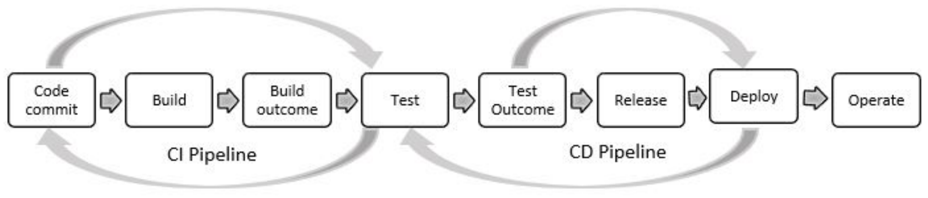


Figure 4.10: CI/CD Pipeline

Tools used for CI/CD implementation differ from organisation to organisation. For example, Finco1 uses the GitLab repository, helm charts to manage their Kubernetes clusters, Docker and Azure container registry for containerization, YAML files for configuration management, Ansible, Terraform scripting, and Istio. Communication is facilitated by internal tools and portals. In Finco2, automated pipelines are created by Azure DevOps. Developers' codes are written in .net or C sharp and committed to a GitHub repository. Sonarcube or Veracode is used to test for vulnerability. They use Docker for containerization and Selenium for web testing. For Finco3 Bamboo is used for CI/CD, they are currently transitioning to Jenkins. Terraform scripting for cluster creation, Ansible for pipeline creation, and Vault for ticket management. Docker Swarm for container orchestration.

Continuous deployment was not mentioned in any of the organisations. In some cases, deployment to production is triggered after manual approval has been given during a sprint. In others, changes are left till the end of the sprint before their deployment to a live environment. DvOps4 described a situation where deployment is done during a sprint or left till its end respectively, depending on whether it is an application or infrastructure code.

"So, I will start with the DevOps engineer. We would basically push this build into a git repository. Um, it would sit there um, until the end of the sprint when we, when we roll out the changes that we've been working on... now for a developer, it's slightly different. Because they would... they have the luxury to deploy there and then." This distinction between deployments of developers and DevOps teams was not noticed in any other organisation.

4.6.2 Collaboration and Knowledge Sharing

In Finco1, the DevOps team1 assisted developers to reconfigure their codes for containerization. DvOps1 said: "And so we joined with the team and we told them how we're actually working. And together with them, we tried to, we need to adjust the application because it was not container-aware. So, together with them, we altered the code a little bit so it was container-ready"[DvOps1]. Knowledge is shared and a mutual understanding of basic activities across the boundaries of teams. This is achieved through a collaborative resolution of code-related challenges. DvOps3 described knowledge flow from developers to DevOps Engineers. "So as I mentioned, there's a couple of proper developers, we have a great resource because they teach us, nope, sit back, look. What are you trying to achieve? And write it properly from scratch rather than just couple together something" [DvOps3]. DevOps teams understand the codes of developers and offer help where necessary. Developers are also made aware of the workings of the automated infrastructure, though not directly involved in its creation or maintenance. According to DvOps1, a level of confidence is brought about by the basic understanding of other aspects of the process and familiarity with the other actors. "And of course, there's the gray area where you need to teach them how the transition from codes to infrastructure works. I can see that whenever it's clear, they would eventually see that ok, we don't have to mind, you know. It's done automatically for us. Which is pretty convenient because we like to write code." [DvOps1] Collaboration is further fostered by teams working closely together, and the willingness to explain one's activities to others. The above is identified as important for collaboration by some other participants. Intra-team collaboration is seen in brainstorming and coding together when issues are encountered. Collaboration in Finco2 involves the DevOps team creating user stories from requirements, breaking them into manageable tasks, and delegating these tasks to developers through Azure DevOps. (if their build is done in the cloud and deployment on-premises, how is this coordinated? who writes and maintains the scripts for the on-premises server such as security, network, etc)

4.7 Benefits of DevOps Adoption

4.7.1 Improved delivery speed and more frequent releases

Participants claim that the implementation of DevOps in their organisation increased their delivery speed and enabled a more frequent release of value. These benefits are aided by embracing continuous practices and automation.

4.7.2 Better Collaboration

Participants report that both developers and IT operations teams (and indeed other teams involved in the software development process) make a conscious effort to align their individual goals. This is because collaboration is one of the core cultural values of DevOps. This collaboration enables building trust between teams and fosters inter-team support to tackle challenges and deliver value.

4.7.3 Software Quality improvement

One result of improved collaboration between developers and IT operations is improvement in software quality. Quality metrics have wider coverage. Continuous testing brought about by automation is said to also contribute to improved quality. Defects are more easily identified, and because visibility is increased in the value stream, solutions to challenges are found faster.

4.7.4 Faster feedback Loop

Participants claim that DevOps has improved the speed of feedback for developers. This is brought about by automation, continuous measurement, and continuous feedback. This fast feedback and collaboration between developers and IT operations enable prompt response to change and accelerate release cycles.

4.7.5 Reliability and Repeatability of Deployments

The practice of automation in DevOps is reported to improve the reliability and repeatability of deployment. This is further strengthened by continuous testing and real-time metrics collection. As automation reduces the possibility of errors from manual processes, releases are more stable and frequent.

4.8 Challenges of DevOps Adoption

4.8.1 Unclear Definition

The lack of a specific definition for DevOps was not categorically stated as a challenge in its implementation. The researcher, however, concludes from the analysis that practitioners' perception of DevOps is instrumental to the benefits or otherwise derived from employing the concept. While some organisations consider it as a job function and focused on its implementation as such, hiring DevOps engineers to carry out DevOps functions, others seem more holistic in their approach, trying to improve the culture of collaboration and knowledge sharing. Thus, varying outcomes are realised which stems from implementation based on individual understanding of the concept.

4.8.2 Changing Technology stack

One challenge mentioned by some participants is keeping up with the fast-changing technology stack and making decisions on the best tools for their specific needs. DevOps teams are constantly faced with the need to upgrade as new versions of tools become available, accruing technical debt during such transitions while trying to meet the requirements of newer versions. As DvOps1 said, "because it's, you know, this movement is just going so fast, these tools exploding but sometimes you just have to make a choice and say, ok we're gonna use this product the upcoming year and we can see afterward, yes."

4.8.3 Undefined and complex skillset

An interesting point noted in the study, which I see as a challenge is that no two participants had the same repertoire of skillset required for a DevOps-aware organisation. This difference apparently originates from differing views of the concept. The complexity of the DevOps skillset makes it difficult to determine re-skilling requirements. Also, learning the tools required for automation and building CI/CD pipelines is considered quite challenging by some practitioners. To enable better collaboration, developers and IT operations teams also needed to train in each other's skills. Some practitioners considered this an unnecessary addition to their workload.

4.8.4 Determination of Appropriate DevOps infrastructure Platform

Another challenge is creating a suitable organisational-wide infrastructure. While it seems effortless to create automated pipelines for a few teams of both development and operations deployments, scaling the platform to accommodate all the varying needs of an entire organisation seems more challenging to the DevOps teams. This oftentimes requires making changes to the infrastructure code and its redeployment. As consequence, the hosted applications must also be redeployed.

4.8.5 Resistance to change

DvOps1 expressed the unwillingness of some development teams in FinCo1 to transform their legacy systems into automation-pipeline-ready chunks. This makes it difficult to scale automation across the organisation. Only greenfield applications and a few willing legacy software teams are currently on the automated cloud platform. Some developers also struggle with the trade-off between infrastructure access and easier deployment.

4.8.6 Requirement changes

For a DevOps team such as DvOps2 that seems to be in direct contact with developers, requirement changes can pose a challenge to optimising the delivery cycle. Although this seems to be more of a challenge for developers. However, because this DevOps team seems to serve multiple purposes such as scrum masters, project managers, infrastructure builders, and writing user stories, requirement changes affect them directly too.

4.8.7 Provisioning hardware in on-premises platforms

The provisioning of hardware sometimes poses difficulty to on-premises platforms. DevOps teams in Finco1 have no direct access to the physical hardware, which is controlled by system administrators. The interface between these two teams is sometimes tricky. “we need to deliver XYZ, and if there are impediments, then we need to look at the escalations which all go through the, eh, PO, product owner...we would use the tools that they have to request new hardware, and we end up back of the queue like everyone else, pretty much.” This challenge can probably be seen as a limitation of on-premises deployment platforms. The team sometimes bypasses official request channels to get their work done faster.

4.8.8 Unidentified Dependencies

Unidentified and unclear dependencies within an organisation and information segregation on APIs were also stated as challenges in DevOps implementation

4.8.9 Poor Code Quality

Some DevOps engineers with infrastructure backgrounds expressed dismay over the poor code quality of their scripts as opposed to that of core developers.

4.9 Summary

This chapter presents a critical examination of DevOps implementation in practice, through an exploratory case study, based on interviews with 11 industry practitioners across nine organisations. Transcripts of interviews were coded and analysed using a method informed by Grounded Theory. Beginning with the description of DevOps, the chapter digs into the approaches taken by organisations to implement the concept. The researcher presents a novel empirical taxonomy of DevOps implementation, describing developers' interaction with On-premises Ops, Outsourced Ops, DevOps teams, and DevOps bridge teams. The taxonomy maps DevOps approaches to on-premises and cloud-based deployments and identified the facilitators of DevOps practices in the different modes. I further identified three distinct groups of activities in the fourth mode: provisioning and maintenance of physical systems, function virtualisation and creation of automated pipelines, and development, deployment, and maintenance of applications, which may have given rise to the implementation of DevOps as bridge teams. The chapter also describes the benefits derived from DevOps implementation and the challenges encountered by participants are presented. Based on my findings, I conclude that DevOps is perceived as both a culture and a job description, and these two views are not necessarily mutually exclusive. Also, the different modes of DevOps implementation seem driven by other organisational factors beyond its perception.

Chapter 5

Strategies for DevOps Implementation: The Roles of Skillset and Automation

5.1 Introduction

Chapter 5 describes findings from the analysis of interview transcripts of phase 2. The data was collected and analysed as described in Chapter 3. The chapter takes the research a step further by providing an in-depth investigation into the pathways to DevOps implementation in organisations and the factors influencing these choices.

Firstly, the methods of diagnostic assessment by the organisations in my study are presented. This is followed by strategies of DevOps implementation. The classification of these strategies (except for platform) is novel and has been published by the researcher (Macarthy & Bass, 2021). The concept of DevOps skillset and its role in DevOps strategies is explored thereafter. This is followed by the role of automation as described by participants.

5.2 Diagnostic Assessment

Participants described various discovery processes used by their organisations. for instance, an organisation used a diagnostic assessment provided by DevOps Research Associates (DORA) as a survey tool. This provided some sort of spatial diagram of the various strengths and weaknesses of the organisation across different dimensions and was used for management decisions on how to proceed with their DevOps implementation. Some others describe some sort of lean engineering approach. [DevCo1_CTO] describes a discovery process where some consultants were invited to workshops with representation from all the different parts of the business (testing, development, database, security, operations, etc). They used techniques like value stream mapping and event storming to identify the most critical constraint that the organisation faced at that time and sought iterative processes of alleviating those constraints using DevOps patterns and practices. DevCo3_MD said "Try and identify where the costs are in their... their work from idea to useful software in the hands of users. And then you can kind of use that as an opportunity to start to improve the quality of feedback usually quite quickly and efficiently or just removing waste from the processes."

Basically, the organisations tried to get an agreement as to what specifically they are trying to improve, where are they now and where are they trying to get to. Once they have the definition of what the specific goal is, the next step seems to be the determination of strategy. Two important things to note here are the fact that practitioners are concerned about the identification of agile or DevOps maturity levels of the organisation and the constraints (mostly in terms of practices, processes, and technology). The constraints that the organisations face very much depend upon the organisation and the industry context.

Some constraints reported by practitioners include instability of their production environment - for which they decided to invest in making it less error-prone, configuration management of services - for which they implemented a configuration management tool, etc. Apart from the initial discovery process, some participants also report continuous diagnostics as part of their continuous improvement efforts.

Once set goals are achieved, the entire diagnostics process is repeated. In DevCo1, the respondent talked of an employee competence discovery process that was used to “put the right people with the right skills in the right teams”.

It raises concern however that only two organisations mentioned the initial discovery process involving employee competency. This suggests that available IT skills competency may not be widely considered a factor in organisational diagnostic assessment for DevOps implementation. The question is how does the available skillset affect the strategy employed by the organisation?

5.3 Strategies For DevOps Implementation

This section describes the strategies employed by the organisations in the study, to implement DevOps. Essentially, this differs from my previous classification of approaches to DevOps based on deployment environment and job functions in the sense that the strategies seem like initial steps taken by organisations in their DevOps journey. The strategies seem to lead up to the approaches described in my taxonomy. The researcher classifies the strategies into 6 groups: Platform, Greenfield application, Monolithic decomposition, Process improvement, Cultural improvement, and Advocacy.

5.3.1 Platform Strategy

Some of the organisations in the study formed teams who developed and managed platforms that allowed the abstraction of software development operational needs onto the platform. This strategy is focused on providing self-service to developers to deploy code on this platform. The researcher found two origins of platform teams. 1. Instances of the technical efforts of this team to implement DevOps practices and increase the flow of change for applications they were already working on. These were mainly cross-functional software development teams, with some level of autonomy. Once this effort is successful and they gain management buy-in, they "morph" into Platform teams, as the infrastructure they create provides services that can be reused by other development teams. The team is consequently charged with the responsibility of evangelising, on-boarding, and coaching other teams. "And we kind of have to just lobby people that we have. They go out to different teams and different managers and tell them, hey listen. You know we have this cool thing. We work with Kubernetes. We would like your team to be part of it." [DvOps1_Finco1].

In one instance, the researcher encountered the intentional transformation of an operations team into a platform team. 2. The organisation brings in consultants who constitute the platform team, with the sole purpose of transforming their organisation into a DevOps-oriented one. This team of engineers (which is a self-sufficient "squad") is embedded within the organisation. The team is responsible for the building and delivery of small pieces of infrastructure to enable Continuous Integration and Continuous Delivery (CI/CD) for select applications. As that relationship continues, they scale up to an organisation level platform team. [DevCo2_lead] describes this as the consultancy model. The self-sufficient "squad" basically provides DevOps as a service for a specific time frame, while training the organisation's engineers to take over. Thereafter, the platform is handed over to the organisation's engineers who continue to provide platform services to other development teams in the organisation. This strategy can be placed in both quadrants 3 and 4 of my taxonomy (Figure 4.5).

5.3.1.1 Platform Infrastructure Engineering

Participants tell us that their organisations begin with a small piece of infrastructure providing just enough services to enable end-to-end delivery of a small application. Basically, the Platform team focuses on automation and creation of the infrastructure part, which is then offered as a service to development teams. This encompasses the lowest levels such as network and security, provisioning application gateway, firewalls, etc.

The platform team is expected to understand and master quite a wide range of tools and technology, to successfully provide the required services to the development team who depend on them. This includes helm charts to manage Kubernetes clusters, YAML files, Azure container registry, Terraform scripting, Istio, GitLab for repository creation, Ansible for deployment, automated pipelines are created by Azure DevOps, SonarQube or Veracode for vulnerability testing, docker for containerisation, Selenium for web testing, Terraform scripting for cluster creation, Vault for ticket management, Docker Swarm for container orchestration and a host of other available tools. Although all the tools mentioned above are not compulsory or used in every single Platform team, the end-to-end delivery process requires a high level of expertise in combining these tools to ensure an easy flow of change. A high level of engineering maturity is described by participants as present in Platform teams.

"so this whole infrastructure stack we create by using Terraform xx in which we can use and deploy infrastructure as code. So I create all the Terraform scripting and modules and I glue everything together. And we also build pre-fab pipelines for our customers so, the pipeline is YAML and everything. So all we want to do, all we want the teams to do actually, write their codes and put it in a repository, hook up our pipeline YAML, and then the application should be container-aware of course, and then it should flow into our platform." [DvOps1_Finco1]

5.3.1.2 Required Skillset and Capabilities

The following are skills and capabilities identified by participants as essential for the successful implementation of platform teams:

1. High network and security skills
2. High Testing and Quality assurance
3. High IT infrastructure architecting and operability
4. High metric and monitoring skills

5. Good development and coding skills
6. Good grasp of a variety of CI/CD tools and technologies

5.3.2 Greenfield Application Strategy

[DevCo3_MD] described a strategy used by their organisation where a pioneering team is selected to begin their DevOps implementation with a new suite of cloud applications. The team was made up of innovative and forward-thinking developers, security and IT operations professionals, pulled from other teams in the organisation. “Cause those teams um... began with more of a clean slate. And so that gives them the opportunity to kind of more aggressively adopting some of the new kind of continuous delivery and DevOps practices.” [DevCo4_CEO].

Here, the newly formed team created and maintained CI/CD pipelines. They were also responsible for the development and deployment of applications. The team was unencumbered with the existing structure of the software development process in the organisation and are rather allowed a great deal of autonomy and innovation. Some interviewees describe this team as a selected in-house team that was highly skilled to carry out DevOps practices. As the processes got established, they began onboarding other applications and extended the practices to other teams. Thus, this strategy can be viewed as experimentation by organisations to establish the values of practices. Greenfield applications strategy was accompanied by migration to the cloud by some of the organisations. The researcher classifies this as leading to the approaches to DevOps described in quadrants 1 and 2 of my taxonomy (Figure 4.5), depending on whether the applications reside in the cloud or on-premises.

Pioneer teams described by practitioners are cross-functional teams with development, security, and IT operations capability. The team is self-sufficient and able to carry out end-to-end value delivery without external dependencies, as they operate outside of the main organisational IT infrastructure. To achieve the purposes of exploration and implementation of new practices, the team is granted a high level of autonomy to make decisions and be innovative. Practices validated by this team are subsequently adopted by other teams in the organisation.

Team members are expected to try out cutting-edge practices to speed up delivery time and decrease downtime. Particularly, participants mentioned continuous practices such as CI/CD, continuous testing, continuous monitoring, continuous measurement, continuous feedback, continuous innovation, etc.

5.3.2.1 Considerations of Greenfield Application Strategy

- Collaboration capability (if the team is not co-located)
- Openness to innovation
- Level of acceptable autonomy

5.3.2.2 Required Skillset and Capabilities

Based on participants' descriptions, the team Greenfield application teams are expected to possess the skills such as:

1. High degree of innovation and independence
2. High decision-making ability
3. High-risk evaluation ability
4. High IT infrastructure architecting and operability skills
5. High development and coding skills
6. Expert understanding of CI/CD
7. High network and security skills
8. High testing and quality assurance skills

9. High metrics and monitoring skills
10. Excellent grasp of CI/CD tools and technology

5.3.3 Monolithic Decomposition

As opposed to the greenfield application strategy, some organisations in the study engage in the aggressive decomposition of monolith applications into modular components. DevCo3_MD described a situation where the organisation had the bulk of its software historically written as a big, complicated monolith. They had not been able to release software into production for five years. As is seen in many situations, the need to increase feedback and delivery speed drives organisations with large monoliths towards modular decomposition.

“One of the first things that [we] started to do is that we started to try and ‘modularize’ the stuff in the organisation. We wanted to speed ... speed up the feedback process. So, we started looking at the technical aspects of that, and we started looking at ways in which we could decompose [our] software into more modular components”

This strategy leverages microservices-based architecture to reduce dependencies and increase the rate of delivery. This is initially done by selected pioneering team(s) for selected products. For example, DevCo2, a company that builds complex scientific instruments (from the operating system firmware and device drivers that run on the devices themselves to sophisticated applications in the cloud, delivering access to the information that the devices generate), had their system based on a huge monolith. Their monolith consisted of many services with dependencies between them, coupled to a single database schema. This made it difficult to make any changes to the application. As this was in a regulated industry and they were required to be regulatory-compliant, they had a culture of enforcing standardised approach to the development and delivery of services, as well as technology. To implement DevOps practices in the organisation, which would help improve the quality and the pace of their feedback, they decided to decouple the monolith and modernise their pre-existing software development practices. The researcher also encountered this strategy in organisations that were in non-regulated industries. Common characteristics found in the monoliths in this study are coupling at the database level and interwoven dependencies between services.

5.3.3.1 Considerations for Monolithic Decomposition

The following were identified by participants as important considerations when decoupling monolithic services:

- Microservices must be aligned to business capability
- Independence of the development and delivery of the microservice
- Team location
- Available skillset

Participants tell us that there is no best method to decouple monoliths, in their opinion. They described the following as ways in which monoliths were decoupled in their organisation. An important point is that the decomposition must be done in such a way that the resultant service structure supports the independent flow of change.

1. Decomposition to achieve specific business capabilities: breaking down the monolith into modules and features that individually address a business capability e.g., pricing, messaging, logging, lost baggage claims, etc.
2. Decomposition according to business domains: Monoliths here are divided along specific domain lines e.g., user management, payment management, etc. A huge collaboration between business experts and technologists is required to achieve this.

3. Decomposition to enable regulatory compliance: In DevCo2, their single application was decoupled in such a way that specific services can still fulfill regulatory requirements without interfering with the flow of change of other services. To enable DevOps capabilities for these regulated services, a participant describes a concept called continuous compliance:

   "they're required to be regulatory-compliant as part of their 'releasability'. So, they're also working on ideas like continuous compliance, so they can automate as much as possible of the compliance function and get feedback um. . . within a few hours or a day ideally".

The teams described by participants in this strategy are cross-functional feature or product teams. This strategy was noticed in hybrid (cloud and on-premises) environments. Here the silos of development and operations were maintained, and the organisations rather worked on the improvement of collaboration and addressing cultural differences between the teams. The teams were however organised around boundaries of decomposition. This strategy is classified under quadrant 2 of Figure 4.5

5.3.3.2 Required Skillset and Capabilities

Depending on the team, participants identified the following skills as required for the successful flow of change in decoupled applications.

Cross-functional teams:

- Design and architectural skills
- Development and coding skills
- Testing and quality assurance
- User interface design

- Basic CI/CD tools and technology
- Database management

Operations:

- Infrastructure management (cloud and on-premises)
- Networking and security
- Metrics monitoring
- Good grasp of CI/CD tools and technologies
- Basic coding skills

5.3.4 Process Improvement

Another strategy seen in the study is DevOps implementation efforts in organisations directed at improving the efficiency of the build system, to address technical problems in the build of their software. It is described by some interviewees as an intentional, proactive approach to improving the overall software development process. This is achieved by identifying bottlenecks in their work from ideas to useful software in the hands of users, improving the quality and speed of feedback, and removing waste from the processes. In this strategy, organisations try to adopt continuous integration and continuous delivery, configuration management, and other process-related practices. The focus is mainly to replace manual processes with automation.

"So, DevOps is about the ability to go quickly and regularly from idea to working software in the hands of users [DevCo2_MD]. Similarly, [DevCo4_CEO] said "So, I talk about it as basically um, the implementation of lean manufacturing and applying principles of lean manufacturing to helping uh, software value streams achieve agility, uh, stability, efficiency, uh quality, security, and satisfaction."

Depending on what the organisation aims to achieve, they carry out a more targeted value stream assessment and start identifying the different stages in the value stream. For example, if it's a lead time goal, and if they define lead time as the time from check-in until ready for delivery, then that defines the boundaries of the value stream, and they look carefully at what is the current time. Time pre-stage and process, wasted time, time to set up between stages, etc. They determine if the contributing factors are people, process, and technology components, that contribute to those timings. They then focus on the relevant part of the value stream to provide specific solutions. Participants describe philosophies such as Six Sigma, CSI, using Kanban boards to visualise, manage and improve workflow and visibility, Lean Manufacturing, and the DMAIC (Define, measure, analyse, improve, control) methods.

The steps to process improvement found in the study are summarised as follows:

- Define the goals
- Identify and analyse bottlenecks
- Identify aspects that will benefit from automation
- Automate identified processes
- Measure and feedback

A popular concept among practitioners focused on process improvement is continuous improvement. This essentially means that the improvement process is carried out iteratively and organisations focus on incremental changes and small achievable goals.

A considerable aspect of process improvement is the modernisation of supporting infrastructure and tool base. There seems to be a focus on things like version control systems and CI/CD tools to improve the quality and the pace of their feedback.

This strategy was observed in organisations with technically challenging and complex software systems. One of the participating organisations built scientific devices in a regulated industry. Their software development consisted of specialist cross-functional teams, centred around specific products.

Participants say that their organisation changed hugely as a result of process improvement. They now have many smaller cross-functional teams than they started with. And they established a deployment pipeline as a service. They are also creating automated deployment mechanisms and feedback channels, different multi-layered tests, and automated testing. This improvement provides feedback to development teams quickly and efficiently. They also provide infrastructure sets easy to generate environments that people can work from.

Participants tell us that for process improvement to succeed, teams need to be given sufficient autonomy and decision-making capabilities mainly around processes to automate. The teams that adopted this strategy are mostly cross-functional, self-sufficient small teams.

"[We] came from a culture of enforcing that kind of standardised approach, and they are now relaxing that a little bit and allowing a little bit more flex...a little bit more autonomy decision-making in the teams."

The skillset for process improvement is similar to that identified for platform strategy.

5.3.5 Cultural Improvement

Some organisations start DevOps implementation by trying to bridge different parts of the organisation and have conversations around collaboration improvements. So, they focus on addressing cultural differences and breaking down some of the barriers. This is especially seen in organisations with geographically distributed teams. [DevCo3_MD] describes an approach of cross-site visitations to enable individualistic relationships and aid better collaboration. Another tactic is the use of local proxies to represent the idea of a product owner. Organisations also employ technical ideas like version control, deployment pipeline, and continuous delivery mechanisms, to provide more global clarity of feedback.

Some participants in this study insist that the strategy of implementing DevOps is in its name: integrating development and operations and breaking down pre-existing silos. [DevCo1_CTO] puts it this way "it is obviously a portmanteau of development and operations… it is removing silos inside the IT department to make the fast (delivery) of work and flow of values em eliminating silos and intensifying the… activities that are focused on customer outcomes and um...output." This is similar to the opinion of FinCo7_Lead, "So, … everybody is in it together and if an application engineer does, em, you know, commits something that breaks or affects the operations engineer, they shouldn't be able to kind of, just wash their hands off and walk away. Em.. so there's a whole kind of philosophical piece to it."

Some organisations start DevOps implementation by trying to bridge different parts of the organisation and have conversations around collaboration improvements. So, they focus on addressing cultural differences and breaking down some of the barriers. Cross-site visitations were used to enable individualistic relationships and aid better collaboration. This is especially seen in organisations with geographically distributed teams. Another tactic is the use of local proxies to represent the idea of a product owner. Organisations also employ technical ideas like version control, deployment pipeline, continuous delivery mechanisms, to provide more global clarity of feedback. It is not to say that these organisations did not employ other strategies along their DevOps transformation journey, but rather that the establishment of better collaboration was their initial focus. This strategy can be classified in quadrant 2 of Figure 4.5.

5.3.6 Advocacy

This basically describes a scenario where a centre of excellence is the unit responsible for working with the development, operations, and product teams in an organisation. The way the transformation worked was getting the right specialist from operations, the right developer, and the right product owner putting them into one group which was then siloed off into what was known as advocates. Essentially, "DevOps specialists" are hired by the organisation to help grow the internal DevOps culture. They work with those different product teams to be able to increase their level of agility and start implementing best practices associated with DevOps in each of those teams. In this strategy, the focus is more on improvements in collaboration. "So, the advocacy model helps those teams to engage properly because in that advocacy model, em, the member of the members of the centre of excellence is not actually doing tangible work relating to those products. They're focusing on the DevOps aspects, so they're able... the centre of excellence work to, to do what they need to in a quick and efficient manner." [Devco4_Lead].

Participants tell us that the organisation must decide on what success looks like at the beginning of the transformation. For continuous assessment, the centre of excellence carried out monthly assessments with the teams they were currently working with. This includes records of what they were up to and what they needed to improve on. They consequently worked with the teams to achieve the needed improvements. This cycle is repeated until the centre of excellence felt that the team was able to carry on and measure themselves. When this level is achieved and the advocacy team feels they could add no more significant amount of benefit, they move on to the next team. Metrics such as deployment frequency, lead time, and velocity (getting an idea of how you are performing as a team DevCO4_Lead) were mentioned in this strategy.

Participants mention software development methods such as scrum, extreme programming, and practices like continuous integration/continuous delivery, continuous feedback, minimal viable products, loosely coupled architecture, minimising hand-offs, delivering small batches, transparency, and eliminating overhead, and testing, using APIs and dedicated teams.

Advocacy is said to provide a third party with a fresh start: "what the advocacy model brings in that capacity for an organisation like that is that the people you bring in have no history of the organisation. Those people are the ones that are more likely to influence the (change) in an organisation rather than the people that have been there 20, 25 years." [Devco4_Lead].

Targets are also easy to evaluate and measure. There are also fewer initial team restructuring and interruptions.

While participants say that the strategy is centred around products, it was not clear what product size is suitable for this strategy.

From participants' descriptions, it can be understood that the Advocacy strategy does not require an initial restructuring of traditional teams. The essence here is to foster and ensure better collaboration between the teams through the intervention of a "third party" (Centre of Excellence). As the COE has an overview of the entire software delivery value chain, they can steer the teams towards an unobstructed and more productive collaboration. Consequently, inter-team interactions and dependencies are better optimised.

"So, the advocacy model works really, really well for organisations that are quite traditionally siloed where they have teams of developers, teams of operations, teams of business analysts, and they got really worked together, and they got teams of architects, and everyone's got very siloed off. So, the advocacy model helps those teams to engage properly"

At some point in this strategy, participants tell us that the centre of excellence tries to implement the "E-shaped professionals" model (having a vertical element of focus on process and frameworks skills, which is important to understand lean, and agile, and value stream management. And then these horizontal strokes talk about human skills, automation, functional, and technical skills) for the organisation. At which time, team restructuring, and topology change is implemented.

5.3.6.1 Considerations for Advocacy Strategy

- Budget availability to hire a specialist team
- Amount of time an organisation is willing to dedicate to the transformation
- Openness to change
- Size and age of organisation: "so, the size of the organisation was a really big factor. Ok. So they, they are a huge organisation with over a hundred thousand employees. An organisation like that, that has a lot of history, old systems, old processes, lots of red tapes to take apart because of the size and age of the organisation, and we really had no choice to use an advocacy model" [Devco4_Lead].

5.4 Skillset And DevOps Implementation Strategies

The skillset required for DevOps is a topic that came up quite often during interviews with participants, mostly in their description of DevOps implementation. Although participants state that they are unaware of any exhaustive list of the skills required to fully implement DevOps in an organisation, they report skills needed to carry out activities such as automated provisioning of environment, with experience in cloud adoption, configuration management, creating automated deployment mechanisms, automated testing, and continuous practices like integration, delivery, testing, feedback, and monitoring as having the most impact in their DevOps journey. Like in every other area of software development, participants report the need for soft skills to successfully implement DevOps on a team or an organisational level. Skills like leadership, emotional intelligence, critical thinking, and strategic thinking were identified as important to be able to clearly articulate the vision and the goals, and measure progress towards that goal and achieve the success of that goal.

The absence of industry accepted skillset for DevOps is seen as a major challenge as organisations have varying requirements to that effect, and practitioners claim not to be quite certain of the necessary skills to acquire.

Participants identified skill gaps as a major bottleneck facing organisations that seek to implement DevOps. In this study, only 2 participants mentioned an upfront evaluation of available skills as part of their discovery process. Some participants report that their organisations realised that there were not enough skills to successfully carry out their chosen approach to DevOps implementation some months into the transformation. Sometimes, the skills required may exist somewhere in the organisation but not in the team implementing DevOps. That would then lead to the problem of team redesign, which requires HR interference and consultation. Other times, the required skills do not exist within the organisation.

[DevCo2_Lead] says "Em.. what I was brought in to do was support the security transformation. However, after about 3 months, it became very clear that their DevOps skillset wasn't mature enough for... to be able to deliver the security transformation, so I ended up spending a lot of my time maturing the DevOps team, before I could then do the actual work that I was tasked with delivering"

Interestingly, participants think that the rate of change in technology and tool-set, especially in cloud computing continues to widen the gap.

Depending on the direction the organisation takes, staff may be trained or self-trained. Consultants also report the informal training on staff. We find this direction to be correlated with the diagnostic analysis at the beginning of the DevOps initiative. The inclusion of skills evaluation in the diagnostic analysis propels organisations toward staff training on lacking skills

5.4.1 Individual Technical Skills Acquisition

Individual IT engineers may sometimes invest their own time and effort in acquiring the skills needed to implement DevOps by paying for their training on e-learning platforms like Udemy. They attend conferences, and meetups and spend time getting acquainted with some toolsets. The acquisition of DevOps-related skills by individual IT engineers is not structured, however, as there is no industry laid out path on what skills are needed at any stage of DevOps implementation. Also, since the organisation is not directly involved at this stage, and there is no structured relevance to the actual roles of these engineers, these skill acquisitions seem rather geared towards individual marketability and career furtherance. Participants also described a situation where challenges in their work drive them to search for technical solutions. This opens up opportunities for DevOps skills acquisition aligned with their functions. We find some of these becoming the pioneering DevOps teams in some organisations. However, even when these skills are acquired, some organisations are not quite ready or willing to implement new ways of working or adopt new technologies. In response to the latter, DevCo1_CTO says "so they go you know, I'm not gonna stay here. I'll go somewhere else, knowing that everywhere else is crying out for these skills, and I'm gonna be very very marketable. .... go on to ... contracting and work for multiple clients or organisations that need implementing and they have the ability to use their best skills." This is seen in organisations more focused on better collaboration and DevOps as a culture rather than on technology.

5.4.2 Organisational Staff Training

As briefly stated earlier, the organisational training of staff can directly be related to skill-related diagnostic analysis carried out at the beginning of the DevOps initiative. Organisations try to fill identified skill gaps by training staff to acquire the skills required to carry out DevOps practices such as CI/CD pipeline creation, configuration management, version control, collaborative tools, etc. According to participants, such training is however dependent on the degree to which the organisation focuses on skills acquisition. Consequently, training is reported to be either minimal - ([DevCo1_CTO] describes it as "**sheep-keeping**") or concentrated. "You take all your staff and you run them through some training program...and they spend one or two days learning the skills. And then, oh you're DevOps certified now, or you're agile certified now. Um.. go forth and be agile." Participants describe this as an ineffective learning technique, as it is difficult to absorb that level of information (particularly with the scope of DevOps) in such a short period. Sometimes, it takes a while before the vision of DevOps transformation is put into action. By which time most people have forgotten what they learned from the training course. On the other hand, practitioners also report the use of coaches who come along and work with the teams and prepare training materials for them. These are most often consultants that help the teams perform better by refining their processes and technology. A lot of time is spent here maturing the skills required to carry out the DevOps initiative in the organisation.

The exclusion of an assessment of skills at the beginning of DevOps initiative, I find, adversely affects the implementation and organisation in the following ways:

- More time spent in achieving goals
- Dissatisfaction of employees
- High turnover
- Initiative abandonment

5.5 Automation In DevOps Implementation Strategies

Automation is identified as an integral part of some DevOps implementation strategies in this study. Based on practitioners' descriptions, I define automation as the use of technology to perform tasks previously done manually. Some participants explained that automation is usually profitable for often-carried-out repeatable tasks in software development. They say this reduces the need for human interference and thus the introduction of human errors.

The identified advantages of automation are summarised as:

1. Time is freed up for more productive work by developers
2. increased efficiency
3. Quality of product/service is increased
4. Error reduction
5. Deployments and service delivery speed is increased
6. Security is enhanced

In the study, automation is observed in areas such as data migration, infrastructure provisioning, testing, configuration management, and orchestration.

5.5.1 Data Migration

This involves the movement of data from one location to another in a computerised manner. Automation here plays the role of ensuring that huge amounts of data can be cleaned, processed, and transferred efficiently and predictably. Cloud migration from on-premises locations was mostly described by participants as part of their organisations' DevOps implementation efforts, in the bid to achieve flexibility, especially with regard to on-premises infrastructure limitations.

5.5.2 Infrastructure Provisioning

Automated provisioning was identified as a key aspect. Participants describe this as a reliable and efficient means to manage access to internal resources such as data, software applications, and IT infrastructure. Basically, this is implemented as Infrastructure as code (IaC). IaC enables the authorisation of multiple user access and resource allocation requests based on predetermined permission levels. Thus, users can quickly access IT infrastructure and data without waiting for long approval lines associated with manual provisioning processes. This also makes the management of the resources easier as changes are made by code.

5.5.3 Configuration Management

Configuration management is described in the study as the capability to keep track of changes in applications, the environment hosting the application, infrastructure, etc, and ensure version control. We find that configuration management spans both development and operation activities. It consists basically of 1) an artefact repository - this is used for file storage. 2)source code repository - containing all versions of code and 3) a database for configuration management. Automating this process ensures that the hosting environment is accurately maintained in an optimal state and resources such as infrastructure, operating systems, configuration files, etc are managed efficiently.

5.5.4 Test automation

Test automation is described as using software tools to manage and test software, to ascertain its expected performance and requirements, before deploying it to production. Most participants report the use of tools such as unit, API, and regression testing tools to achieve test automation. They claim that automated testing increases accuracy and improves the detection of bugs as well as usability. Overall, practitioners tell us that automated testing contributes immensely to their team's agility and product quality.

5.6 Summary

This chapter gives an in-depth exposition of the strategies of DevOps implementation in the study. The researcher presents a published novel classification of the strategies (Macarthy & Bass, 2021), identified as platform, greenfield application, monolithic decomposition, process improvement, cultural improvement, and advocacy. The roles of skillset and automation are also exposed in the chapter; however, they will be further discussed in the Discussion chapter. These factors, along with others identified subsequently in literature, will be employed in the creation of a model for DevOps strategy determination.

Chapter 6

DevOps Strategy Determination Model Creation

6.1 Introduction

This chapter examines the findings already presented in this thesis and demonstrates the formation of theories. Based on the theories generated and further findings from literature, a model for the implementation of DevOps is created. The next section critically examines the roles of skillset and automation in DevOps strategies. Thereafter, theories formed are given. Then the model creation process is shown, culminating in the presentation of the model itself and its description.

6.2 An Examination of the role of Skillset in DevOps Strategies

Although skills are required for almost any task in software development, our findings indicate a striking sort of correlation between the strategies adopted by the organisations to implement DevOps and the skillset. In the case of the pioneering in-house teams in the platform strategy, the teams acquire the skills required to successfully carry out DevOps practices, which they utilise to build platforms for themselves. With management buy-in, these platforms are extended outside the team. It clearly shows organisations leveraging available skills to carry out DevOps transformation. The engagement of consultants to constitute a platform team and lead the transformation by evangelising, on-boarding, and coaching teams signifies the absence of the required skillset in the organisation.

This is also an indication that other factors influence the choice of an appropriate strategy for an organisation beyond skillset. However, participants from organisations with in-house platform teams claim that the organisations went the way of platforms largely because their team already had the capability. This is similar to the greenfield approach, except that here, the team is constituted by the organisation to deliver brand new applications in the cloud using DevOps practices. This team serves as a model to evaluate the changes to follow. In a way, organisations can reasonably envisage the outcome of the changes they wish to implement following results from the model team.

In DevCo2, even though they had adopted the monolith decomposition strategy primarily because of their set goal, they soon realised that the required expertise was absent. “The team that I mentioned, the one driving the changes, saw the lack of ‘DevOps skills’ as being a key barrier. The tech in the org had stagnated a bit and they saw the adoption of what they thought of as ‘agile and DevOps tools’ as being a big step forward. So, they hired and trained to bring in skills in new tech, Git, JIRA and later Docker and Cloud tech.” [DevCo2_MD]. Cultural improvement and advocacy strategies show the least dependence on pre-existing DevOps skills. The organisations are rather focused on improving collaboration. “to do things like implement minimum viable product and implement the loosely coupled architecture... You know, architecture sits with solutions architecture and enterprise architecture. Em, minimum viable product sits with product management, so there are two separate teams that you need to engage with and bring on that same journey toward … to be able to get to where you want to be”.

Even DevOps enthusiasts without the required skills will need to be trained, depending on the choice of strategy. Participants in this study described how the non-inclusion of skill identification at discovery resulted in delays in the achievement of goals as focus shifted to upskilling further down the line. In the same vein, strategies that overlook skills already acquired by employees might lead to the migration of those employees to places where their skills will be put into good use.

We believe the following skill-related questions are pertinent for choice of an appropriate strategy for an organisation: Are the DevOps skills for the strategy available in the organisation? Would the strategy require hiring of professionals with the required skills? Are there budgetary constraints to acquiring these professionals? Would immediate upskilling be required to facilitate the strategy? How much team re-design would be needed for such strategy?

6.3 An Examination of the role of Automation in DevOps Strategies

6.4 Theories Generated from Data Analysis

Following Grounded theory process of systematic and detailed data analysis, this research arrives at the theories below. Theory is used here in the sense of the theoretical outline generated as a result of sorting and coding (Hoda et al., 2012)

1. A striking correlation exists between the skillset and the strategy adopted by organisations to implement DevOps. Thus, the choice of organisational DevOps implementation strategy should generally be based on a consideration of technologies, processes, culture, and skillsets as top-level concerns.
2. The inclination of an organisation towards upskilling is relevant in the determination of the DevOps implementation strategy to adopt.

3. Automation seems like a key enabler of some DevOps implementation strategies. Automation should be structured and planned out.

4. Metrics and continuous measurement are critical to a successful implementation of a DevOps strategy

These four theories contribute to the design of component parts of the DevOps strategy determination model, which will be presented and discussed.

6.5 Devops Strategy Determination Model Creation

A critical examination of the DevOps implementation strategies identified in this study shows observable differentiation in concepts like team structure, skillset, project type, responsibilities, and intra-team collaboration and autonomy. While I do not claim that this is an exhaustive list of differences in the strategies, I focus on these areas as the analysis of data collected bring them to the fore. Within each strategy, I noticed that the software development method employed cannot be clearly differentiated. This means that variants of similar methods were described by participants in different strategies. For example, agile practices like pair-programming, daily stand-up meetings, retrospectives etc were mentioned across the strategies with no apparent link to the characteristics of the strategy beyond "tailoring" for specific projects. Figure 6.1 illustrates my understanding of the occurrence of software development methods within DevOps implementation strategies.

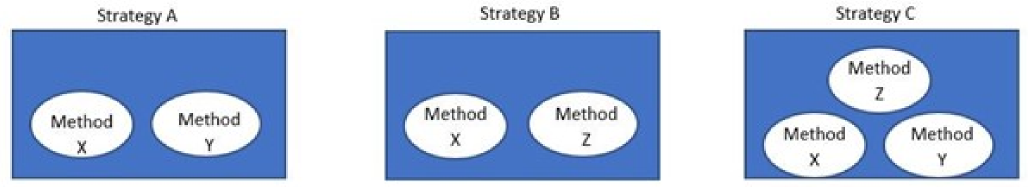


Figure 6.1: Methods in DevOps Strategies

6.5.1 Similarities between DevOps Strategies and Method Engineering

Some of the characteristics of “method” described by these researchers are similar those of the DevOps implementation strategies previously mentioned:

1. The strategies are all specific approaches or processes to carry out a system development project.
2. The way of thinking of each strategy is clearly identified and different from others
   (a) Platform strategy focuses on creating platform-as-a-service to other teams in the value chain.
   (b) Monolithic decomposition strategy is directed towards the decoupling of huge monolith into loosely coupled, easily managed services.
   (c) Greenfield application strategy focuses on the implementation of DevOps using a brand-new application in new environment, unfettered by the restrictions of an existing one, providing a lot of autonomy and room for experimentation by select team.
   (d) Process improvement focuses on identifying and removing waste in the overall value chain while implementing practices to improve the flow of change.
   (e) Cultural improvement aims at improving collaboration between various teams on the software delivery pipeline.
   (f) Advocacy is the use of “centres of excellence” to sell the idea of better collaboration and DevOps practices to individual software development teams in the organisation.
3. The structure of development activities in each strategy can be clearly identified and differentiated.

4. Just as a method is not applied in a textbook prescribed manner in practice, the strategies described in this study may not be found in the same ways in organisations outside of the study.

From both definitions previously cited, methods have well-defined directions and rules. The DevOps implementation strategies can therefore not be directly classified as methods. However, Figure 6.1 shows that methods are subsections of the strategies. Also, the strategies share a distinctive feature with Situational Method Engineering (SME): a focus on the project at hand. The identified characteristics and distinguishing features of DevOps strategies provide foundation for the design of a model for strategy determination.

Considering the similarities between method and the DevOps implementation strategies, I propose the adaptation of Brinkkemper (1996) structure of SME to the determination or creation of DevOps strategy for an organisation. This study therefore leverages Brinkkemper's structure of SME and extends its principles to account for specific characteristics of DevOps implementation strategies, according to my research findings.

6.6 Introducing SIDSEM – A Situational DevOps Strategy Engineering Model

Figure 6.2 shows SIDSEM. The light blue arrows and boxes show the main modifications made to Brikkemper's model, following the findings of this study.The theories stated in section 6.4 contribute to the development of the model in the following ways:

Based on Section 6.4 (1), I have separated skillset from the general factors for characterising a project. My analysis uniquely identifies the significant role of skillset in the various DevOps strategies. To determine a suitable strategy, an individual comparison of selected DevOps strategy fragments with existing skillset is required. This ensures an ability to fulfil the bespoke strategy when assembled.

In accordance with my second theory (Section 6.4 (2), the inclination of an organisation towards upskilling is identified as contributory factor to the selection of suitable strategy fragment. This is presented as “training” in the model. Strategy fragments can thus be chosen despite the lack of skillset for fulfilment if the organisation is able to acquire the necessary training.

Following my third and fourth theories [Section 6.4 (3 and 4)], automation, metrics and continuous measurement are presented as key components of the model. These will be described shortly.

6.7 Component parts of Proposed model

The model presented in Figure 6.2 consists of 4 basic components:

1. Strategy administration and strategy base
2. Selection and assembly of strategy fragments
3. Process automation
4. Metrics and continuous measurement

These components are shown in Figure 6.3.
The Strategy administration and strategy base, and Selection and assembly of strategy fragments stem from the comparison made in section 6.5.1. However, methods have been replaced with DevOps strategies in my model. Process automation has been added to the original model, as it is an enabler to some DevOps implementation strategies. Metrics and continuous measurement differs from the original model in the sense that teams do not wait till the end of the project to measure the impact of changes made to their processes. Metrics and continuous measurement is an encouragement to teams to proactively and constantly use collected metrics to evaluate their processes.

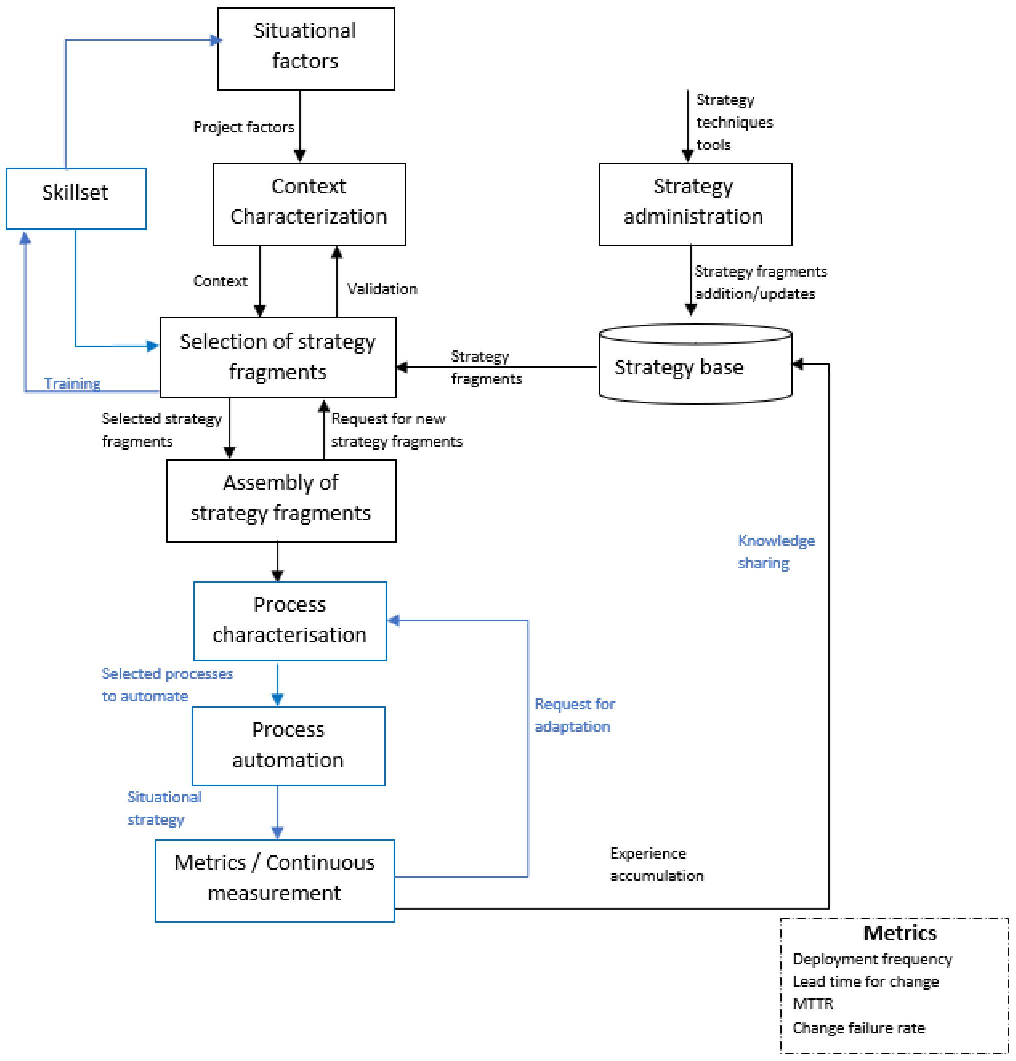


Figure 6.2: SIDSEM – A Situational DevOps Strategy Engineering Model

6.7.1 Strategy Administration and Strategy Base

This component part comprise of:

1. Strategy administration – DevOps implementation strategies are selected from literature (findings from this research) and broken down to fragments

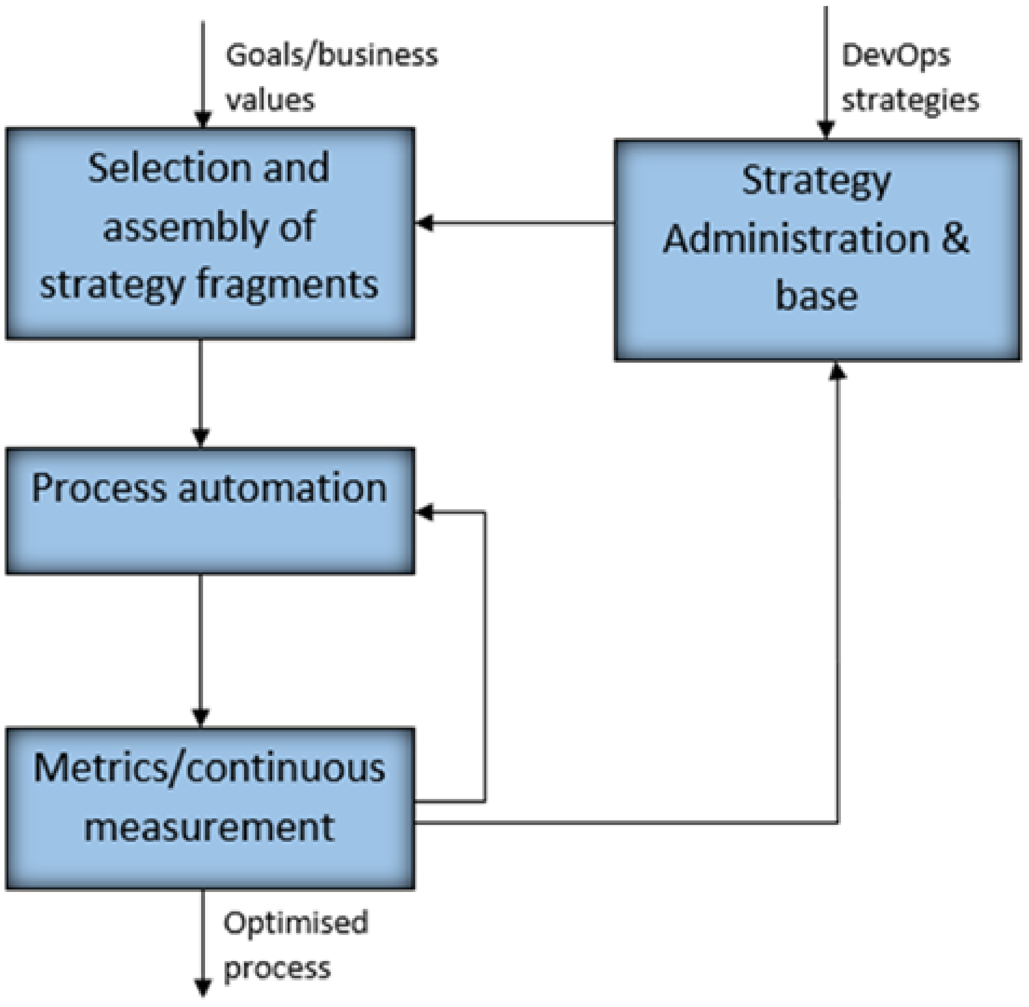

Goals/business values
DevOps strategies
Selection and assembly of strategy fragments
Strategy Administration & base
Process automation
Metrics/continuous measurement
Optimised process

Figure 6.3: Model Component Parts

of products and processes following prescribed metamodels. The Method Engineer (this could also be anyone responsible for software development processes)

- Identifies DevOps implementation strategies from literature.
- Selects suitable strategies – Our study identified 6 strategies in practice. The Method Engineer is however not restricted to these.

2. Strategy base – Method base is described as a "repository" of method parts (Brinkkemper 1996; Harmsen 1997; Rolland et al. 1998). According to

Henderson-Sellers, method parts are “small portions of a methodology, either a methodology that already exists or a methodology to be formed”. They insist on the standardisation and conformity of each part to “a higher-level definition given in a metamodel”. Comparing this to the DevOps strategies identified in my study, I refer to the strategy base as a repository of fragments of the six strategies, alongside knowledge and experience gathered during strategy administration, continuous measurement, and knowledge sharing.

I describe DevOps strategy fragments are small parts of the identified strategy that represent recognizable aspects of it. This aligns with Ter Hofstede and Verhoef (1997) definition of method fragment as – “a coherent part of a metamodel, which may cover any of the modelling dimensions at any level of granularity”. To determine and represent the fragments of the DevOps strategies, I ask the following questions:

- How do we divide the strategies into reusable/adaptable fragments?
- How will the fragments be characterised?
- How will the fragments be coupled to form a situational strategy?

To answer these questions, I follow the principles of ISO (2014) - an international standard definition of a meta-model for software development methodologies, and the Perspective dimension of meta-modelling for situational method engineering proposed by Brinkkemper (1996). I describe the process-focused meta-model as shown in Figure 6.4
Our product-focused meta-model is described in Figure 6.5.
A metamodel is a structural description of tasks (processes), techniques, work products etc (Brinkkemper, 1996). The metamodel guides the Method Engineer on how to break down the selected strategies to reusable fragments - specific processes (e.g., new pipeline set up) or products (e.g., automation log). Based on the data analysis and the meta-models I have described in the tables above; I give examples of product-focused and process-focused DevOps strategy fragments. The capability

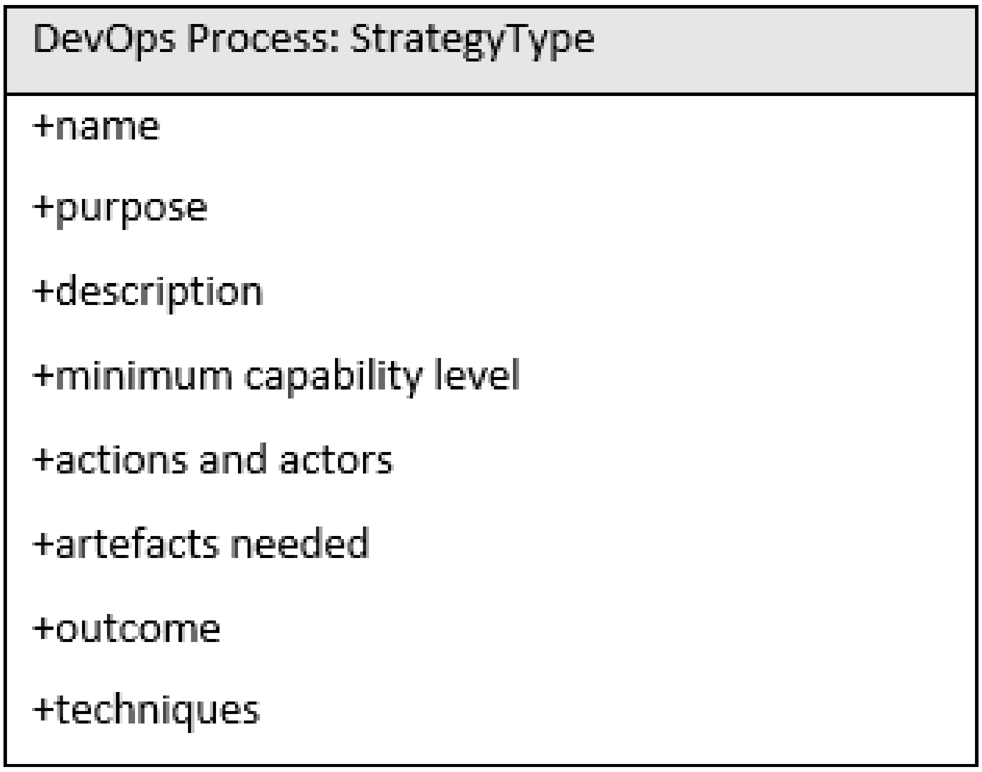


Figure 6.4: Process-focused meta-model

levels follow the definition of Society (2012), the global skills and competency framework for the digital world. The Method Engineer creates the fragments and stores them in the strategy base. Table 6.1 is an example of a product fragment.

This fragment can however be further broken down to smaller bits such as design version control, integrate source control, implement and manage build infrastructure, implement code flow, implement continuous integration etc. A process fragment is described in Table 6.2

6.7.2 Selection and Assembly of Strategy Fragments

Fragments are selected from the strategy base (depending on the organisational context - goals/objectives, skillset etc.). Selected fragments are assembled depend-

Table 6.1: Product fragment

NewPipelineSetup: Platform

Name: New Pipeline Setup
Purpose: Create a channel to ensure the seamless movement of codes/software solutions from development to deployment and monitoring. This may be based on new business requirement, platform migration, or cost saving.
Description: The integration of tools and automation of processes (for core infrastructure deployment or configuration of core services) to develop, test, and deploy new codes rapidly.
This is an automation of the software delivery process by using connected tools, which eliminates the need for manual intervention. It ensures a reduction in software delivery time as well as potential human errors, resulting in high quality codes, faster and more frequent deployments. This task requires the combination of continuous practices to ensure the various aspects of the pipeline are properly integrated to provide the expected seamless delivery. One of such practices is continuous integration and continuous delivery (CI/CD). This ensures that multiple developers can independently work on small parts of a new code and integrate it in a shared repository, automatically test for bugs, deliver bug fixes, and release new features at high frequency.
Another essential practice for this task is continuous monitoring, which provides real-time overview of the organisation's IT infrastructure and services to both operations and security teams (or other relevant teams, depending on the team structure), and aid the provision of real-time support for critical processes.
Minimum capability level: 2 - proficient in a programming language, extensive knowledge in system administration, distributed systems.
Actions:
Creates a delivery pipeline by:

- Selecting suitable CI/CD tools
- Establishing source control for conflict-free collaboration
- Setting up suitable build server
- Including automation tools for testing

Artefacts needed
Business requirements document
Outcome
• A delivery pipeline is set up
• Any other artefacts produced e.g documentation
Components: • CI/CD
• Continuous testing
• Continuous monitoring
• Continuous measurement
• Continuous feedback
Techniques:
• Infrastructure as code
Actors:
Product owners, IT Operations, IT Security, Developers (depending on the team structure), DevOps team (depending on the team structure).

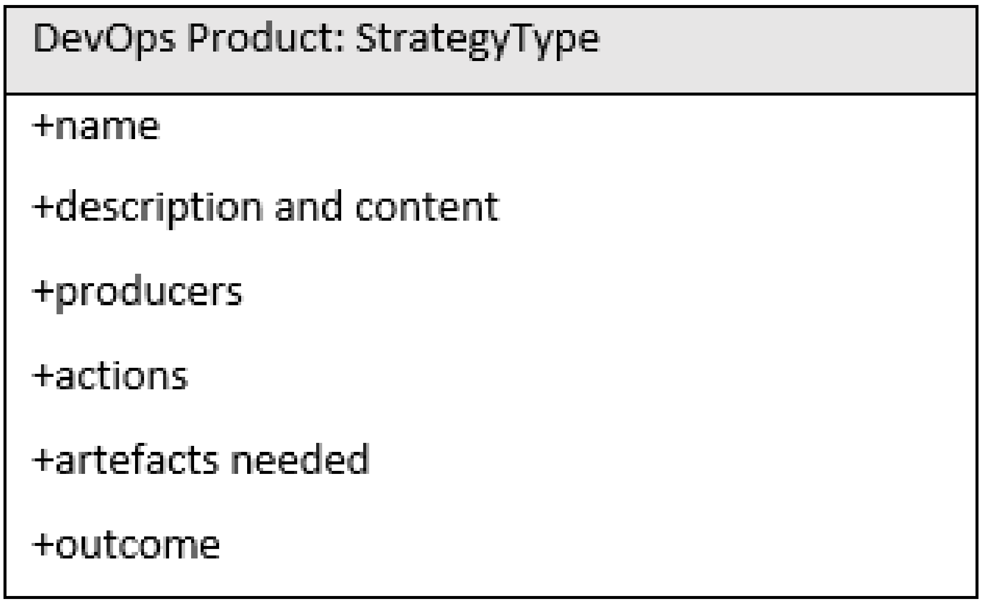


Figure 6.5: Product-focused meta-model

ing on their technical relationships. To further describe this segment, I divide it into project environment and characterisation, selecting appropriate strategy fragments, and assembling selected fragments :

6.7.2.1 Project environment and characterisation

The project environment describes situational factors of both the supplier and the customer organisations, grouped in facets. Figure 6.6 shows situational factors identified in literature:
The Method Engineer should:

- Identify the relevant project / organisational factors (from situational factors.
- Classify each factor as low, normal, or high (depending on the level of involvement of that characteristic).
- Determine the relevant facets based on high involvements.
- Define the project context.

Table 6.2: Process fragment

AutomationBacklog: Platform
Name: Automation log **Description and content:** the key aspect in any devops team is automation. Business processes (in the context of infrastructure deployment and configuration) can be automated and as a result, each time a process runs it will generate a log describing its status, and errors. **Producers:** Logs are generated by computer processes and traditionally were stored on flat files. With the advance of distributed systems, logs have increased exponentially and are stored and processed for further analysis in central databases where engineers can investigate particular incidents. **Actions:** When processing logs, the timestamps integrity needs to be guarded. The rest of the log message may be transformed to fit a certain standard. Artefacts needed: A logging pipeline, and a log database

These steps are shown in Figure 6.7

6.7.2.2 Selecting appropriate strategy fragments

The Method Engineer should:

- Compare the context with the strategy fragment description to determine suitability
- Compare the required skills of the strategy fragment with the skillset to determine availability
- Determine the possibility of skills training where required
- Choose most applicable strategy fragments

6.7.2.3 Assembling selected fragments

building custom strategy from fragments.

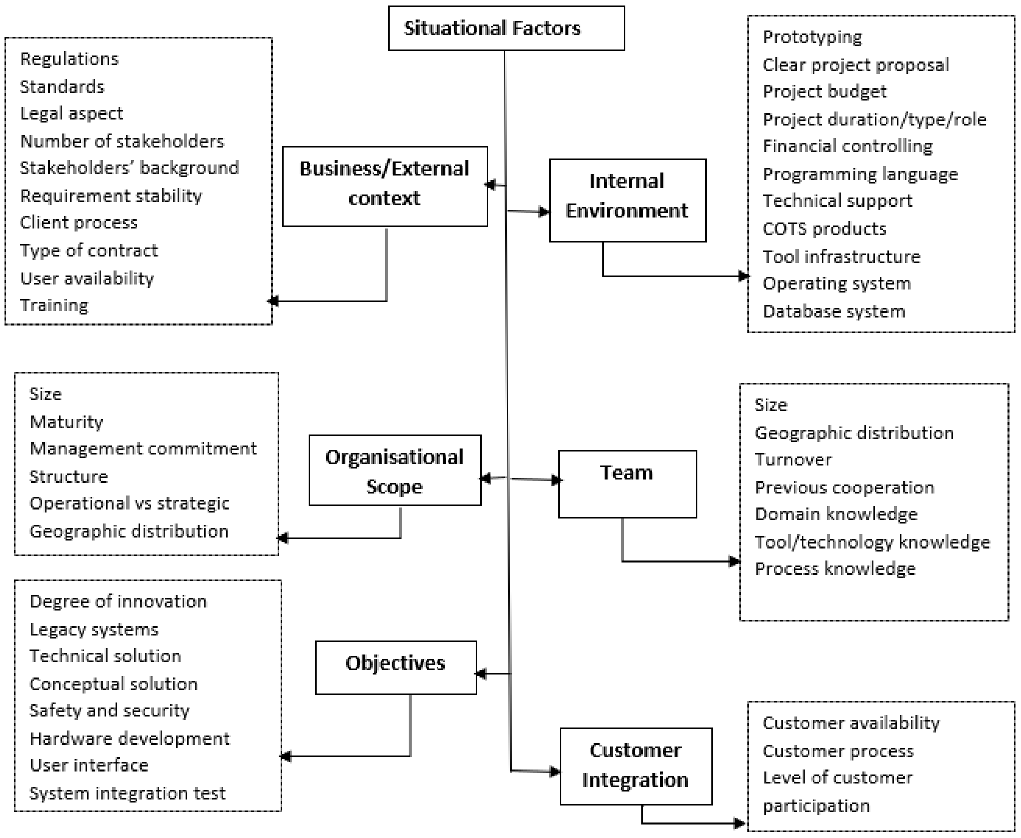


Figure 6.6: Situational Factors - adapted (Clarke & O'Connor, 2012; of Government Commerce, 2002; Slama et al., 2015; Firesmith & Henderson-Sellers, 2002; Kornyshova et al., 2007; Kalus & Kuhrmann, 2013; Giray & Tekinerdogan, 2018)

Many approaches exist to assemble method fragments in Information system development. We suggest assembly by association and assembly by integration, for simplistic useability. The method engineer should evaluate the technical relationship of the chosen fragments.

Assembly by Association: This should be used when the chosen strategy fragments have different system engineering functionalities/goals.

- Identify Links between the concepts

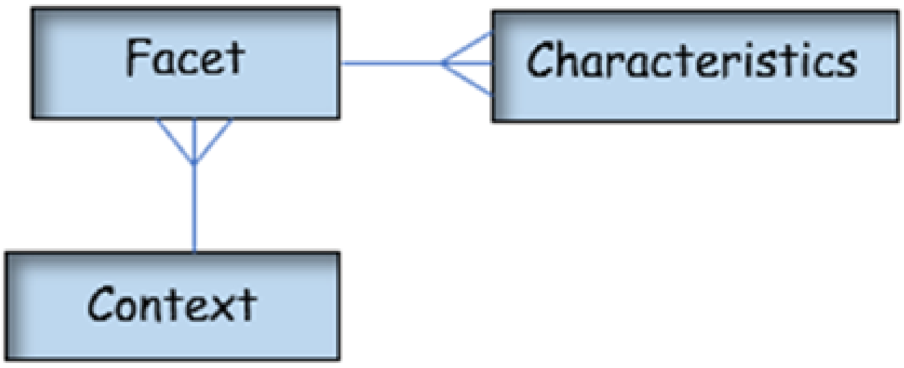


Figure 6.7: Project Characterisation

- Fragments should be ordered in complimentary fashion. E.g. The outcome of one fragment could be input to another fragment

Assembly by integration: To be used when the chosen strategy fragments have similar system engineering functionalities/goals.

- Identify and merge common elements
- Fragments should be ordered in complimentary fashion

6.7.3 Process Automation

This component part of the model comprise of automation candidate selection and process automation guidelines

6.7.3.1 Automation candidate selection

This is the identification of candidate processes for automation in the customised DevOps strategy. It includes an upfront evaluation of the benefits of such automation and the determination of the impacts on specific metrics and business goals. Characteristics of IT processes for automation include:

- High volume transactions
- Time-sensitive transactions
- High value transactions – impact of error
- Frequent access to multiple systems
- Limited human intervention
- Limited exception handling
- Process prone to error or rework
- Ease of decomposition into clear IT processes
- Clear understanding of manual cost

6.7.3.2 Process Automation Guidelines

After selecting the processes to automate, the following steps are recommended:

1. Identify the process owners and other stakeholders – clearly understood workflow.
2. Create automation plan –steps involved as well as possible outcomes
3. Estimate time needed to automate the process

4. Choose the right tool – evaluate existing technology stack to determine a suitable and easy-to-integrate tool.
5. Set up change management plan – e.g staff training and tutorials for those interacting with the process.

6.7.4 Metrics Collection/Continuous Improvement

The organisation determines the metrics to collect based on the goals pursued. The collected metrics are used to aid continuous improvement, which could include making changes to the process and implementing other DevOps strategy fragments. We recommend the following steps for continuous improvement:

1. Collect pre-determined metrics
2. Analyse the metrics to determine impact of the changes made to your process
3. If change is successful, record outcome and experience accumulated in the strategy base. Else, go to step 6
4. Plan knowledge sharing of experience
5. Identify other DevOps improvements to make repeat the model
6. If change is unsuccessful, re-evaluate the processes to determine required modifications.

6.8 Summary

This chapter presents to the reader the steps taken to create a novel DevOps strategy model based on findings in this study. The model is intended to provide guidance to organisations both for a successful DevOps implementation, and continuous improvement. The component parts of the model are described and prescribed steps clearly explained.

Chapter 7

DevOps Strategy Engineering Model Evaluation

7.1 Introduction

This chapter conveys to the reader the steps taken to evaluate the DevOps strategy engineering model – SIDSEM. Firstly, a focus-group workshop with expert practitioners of the participating organisation is presented. Thereafter, the findings from the workshop are described. Quotes are used from the workshop to demonstrate participants response to the model. A call to action follows, consequent on the findings. The proposed action involves the implementation of parts of the proposed model relevant to the context of the participating organisation. The chapter concludes with the instantiated repository of SIDSEM, created in response to participants suggestion. It aims to further improve the practical usability of SIDSEM. According to Robson & McCartan (2016), "evaluation is often concerned not only with assessing worth or value but also with seeking to assist in the improvement of whatever is being evaluated". **This is Phase 4 of the study**.

7.2 SIDSEM Focus Group Workshop with DevCo9

The workshop consisted of two main sessions. First, I gave an overview of the study which was mainly driven by the question of how to successfully implement DevOps in an organisation. To answer that question, I presented 3 focal steps in my research: 1. Identify existing patterns of work coordination in DevOps implementations 2. Investigate the patterns that influence such patterns 3. Develop a model to aid appropriate implementation of DevOps. The research findings were then presented. This included the taxonomy of DevOps approaches, the strategies of DevOps implementation, and a mention of SIDSEM. Before the introduction the model, several starter questions were discussed. These questions were aimed at information gathering of the current state of their software development process and practices. 1. How would you characterise your project environment? 2. How do you determine DevOps / Agile practices to implement? 3. How do you decide on processes to automate? 4. What metrics is most relevant to your business goals? Each question was explained clearly, with simple examples, to eliminate ambiguity.

The second part of the workshop consisted of the presentation of the model by the researcher, and the evaluation of DevCo9 software development process with the model, by the Focus Group. The researcher clearly described its component parts and implementation process. The evaluation was a comparison of the their existing processing with the components and proposals of the model. Thereafter, the participants had an in-depth discussion on the model's potential to improve the agility, operational efficiency and reliability of a software development team. The engagement with DevCo9 is summarised as shown in Figure 7.1.

7.3 Focus Group Workshop Findings

Findings from the evaluation of DevCo9's software development process with SIDSEM is presented in this section.

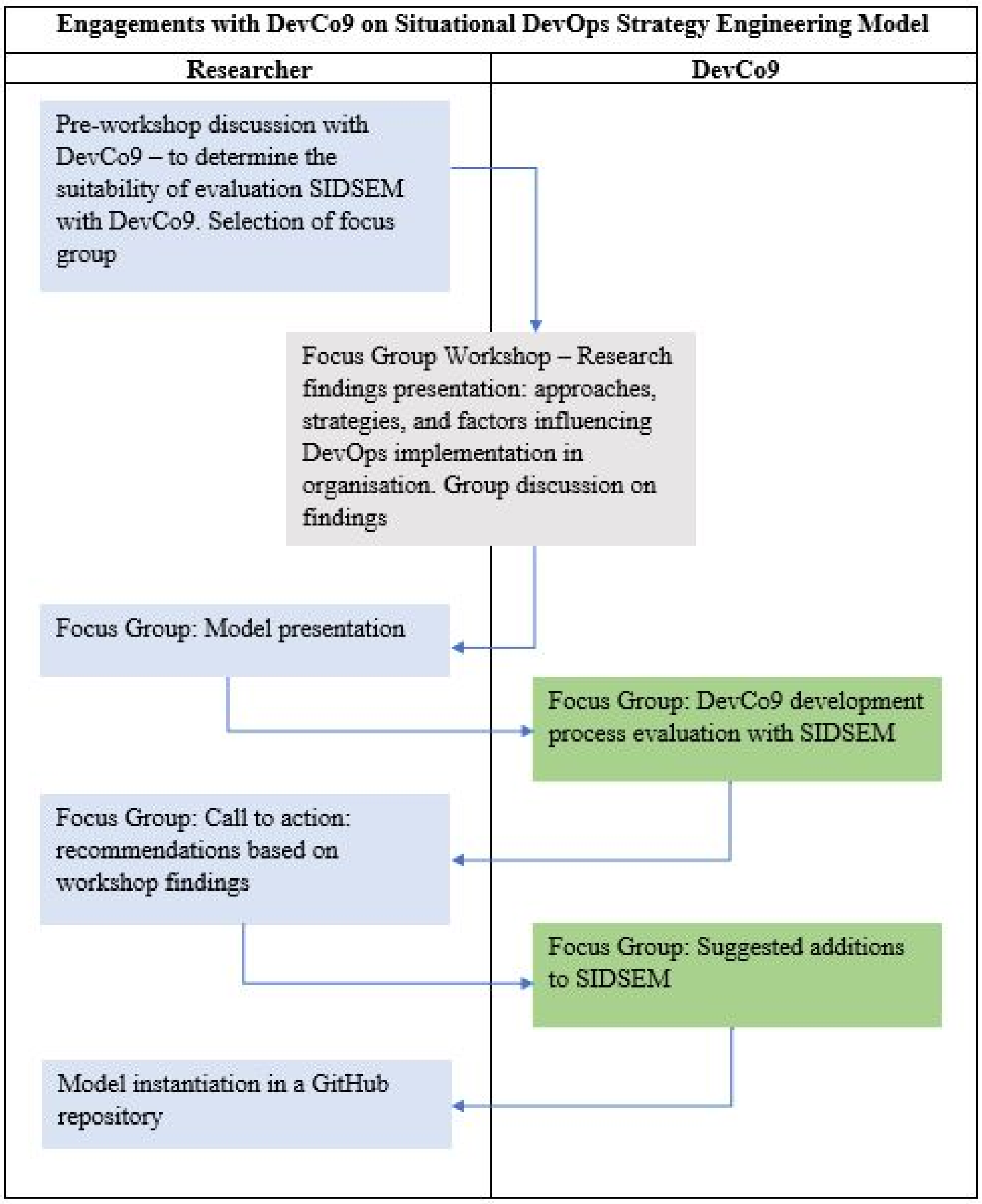

Engagements with DevCo9 on Situational DevOps Strategy Engineering Model
Researcher
DevCo9
Pre-workshop discussion with DevCo9 – to determine the suitability of evaluation SIDSEM with DevCo9. Selection of focus group
Focus Group Workshop – Research findings presentation: approaches, strategies, and factors influencing DevOps implementation in organisation. Group discussion on findings
Focus Group: Model presentation
Focus Group: DevCo9 development process evaluation with SIDSEM
Focus Group: Call to action: recommendations based on workshop findings
Focus Group: Suggested additions to SIDSEM
Model instantiation in a GitHub repository

Figure 7.1: SIDSEM evaluation engagement with DevCo9

7.3.1 Pattern OF Work Coordination

In relation to the DevOps taxonomy previously presented in this study and presented in the workshop, the Focus Group identified the approach to DevOps in DevCo9 is outsourced Ops. However, the organisation also has several on-premises legacy applications which was managed separately by an Ops team. The DevOps strategy employed is Greenfield application. Newer applications are cloud-native. They are developed and managed using tools like Docker and platforms like AWS. The development teams are mainly expert developers familiar with. The team seem to be highly innovative and self-motivated to learn new ways of delivering quality service to their clients. The team also involves in continuous knowledge-sharing of newly acquired software development skills. This is most times a formal, planned meeting. Knowledge sharing also happened informally during brainstorming sessions.

“Yeah, so I think our teams, mostly developers, so anything we do would be outsourced operations. So, it’s developer outsourced OPS... We generally just start from scratch because our legacy stuff just isn’t, we can’t really upgrade the legacy stuff to do what we kind of need it to do.” [DevCo9_TechLead].

”And the choice of software practices is based upon how we feel that the software industry is currently going at the moment and where we think it’s heading. If we feel that it will benefit us yeah. We definitely do some training around that.”[DevCo9_Dev1]

7.3.2 Project Characterisation

Projects usually originate from the business and are evaluated from the business point of view. To characterise their projects, the team does a requirement analysis then puts it forth for approval, based on available budget. A major consideration for projects is the technical requirements. The team also identify the skillset required for the project to determine its feasibility.

“Because they’re coming from a traditional consultancy firm. They’re not the technical side of it isn’t really demanded from them, so it’s they don’t really understand exactly how the tech works. So, in a sense, that gives us a little bit of freedom to be able to go back and say this would be the best way to do it, and this these are the reasons why, so we can kind of lead that.” [DevCo9_TechLead].

“How do we determine what our agile practice is simple, and it’s a good question. I haven’t got a clue. All of them are maybe reactive.” [DevCo9_Mgr]

“Yeah, well, I think I’ve. I don’t know if we’d say well where we active. I think we’re proactive. Like if you look at our solutions, we do have like a lot of documentation around them. I was just reading about Kubernetes and I was like, oh, this sounds interesting. This would really help out [DevCo9]. I went away and played with it and got it to a point where I could understand it. And then I showed it to horse and then we decided to move forward with it.”[DevCo9_TechLead]

7.3.3 Automation

Automation of processes in DevCo9 is reactive as opposed to it being planned and structured. The development team had weekly backlog review, like a sprint, and evaluate priorities, depending on what is on the backlog. Although, automation is not the focus of these reviews, the decision of the team to automate processes may arise depending on the perceived user needs and the bid to increase speed of development.

“Uh, currently I decide on which processes to automate based on the amount of emails that I get.” [DevCo9_TechLead] Balancing between time spent in automating processes and delivering expected business value is a challenge. The workshop participants agreed on the difficulty encountered by the team when trying to automate processes and meet up expectations of solution delivery at the same time. “ We try and get some kind of automated process, [but] we obviously we have limited time. But it would be alongside their tasks.” [DevCo9_Dev1]

7.3.4 Metrics and Continuous Measurement

Metrics in DevCo9 is mainly around the performance of their applications. It involves investigating incidents by reviewing system logs. The efficiency of their processes or outcome of changes are not measured by any metrics.

”So, we don’t monitor velocity or anything like that, that’s another weakness, so, that’s something we can we can certainly improve on but the metrics we have is mainly on database side…how many active connections we have, what throughput is, how many request in every second, what percentage of cpu usage is being utilised, how much storage is being utilised. The performance metrics are what provided us by the service we use. We’ve not created anything ourselves yet that will help us monitor how we’re performing around our applications.”

7.4 Discussion on the model: Focus Group View

7.4.1 Relatability

DevCo9 did not currently use a formal model to structure their software development processes. However, the Focus Group agree that the component parts of the model was relatable to them.

“So I think this is brilliant. I don’t know. Honestly, I think it’s amazing… I can, I’m looking at this now. I can understand I’ve actually been involved in every single one of these boxes and I could give an example for each and every single box and and know exactly what I did. I just did it without knowing that these boxes existed”[DevCo9_Dev1]

7.4.2 Suitability

All the participants agreed that the model would be suitable for the improvement of their software development processes.

“I think it could be applied pretty well too. So what was it? Definitely areas of it can be applied pretty well to some of our processes. Regarding iterative steps of selection, strategy, fragments and assembly of strategy fragments, we kind of do that in it’s a bit. It’s about, yeah, yeah, we could do aspects of this, and I think definitely regarding selecting the processes to automate that would help. . . . Like the selection of strategy fragments and then falling back to training to improve the skill sets and the application of the appropriate skill sets for the different strategy fragments.” [DevCo9_Dev1]

7.4.3 Clarity

The model presentation initially seemed intricate to some of the participants. The researcher attributed this to the fact that this was not a fully-mature DevOps team. To reduce the perception of complexity, the team was given the simplified DevOps Implementation model. This version contained the basic parts of the model with a full description of the activities and guidelines to implement them, which seemed better understood by the practitioners.

“Uhm, well, it seems a little bit complex at first when you first look at it so. But it may be daunting for some organisations when they first look at it. When I first saw it was. But once you explained, I think it’s a bit more clear” [DevCo9_Mgr]

7.5 Call to action: Model implementation Recommendations

The findings from the workshop demonstrate the while there is no formal model for DevOps implementation in DevCo9, the expert team had intuitively carried out the activities in components 1 and 2 of the proposed model. Components 1 and 2 are quite similar with the original Brinkkemper's model. The major difference being my focus on DevOps implementation strategies where the previous model was concerned with software development methods. However, the information gathered from the workshop also show a lack of a clear structure of process automation and continuous measurement. These make up parts 3 and 4 of my model. Figure 7.2 shows the proposed actions to help DevCo9 better improve their DevOps implementation.

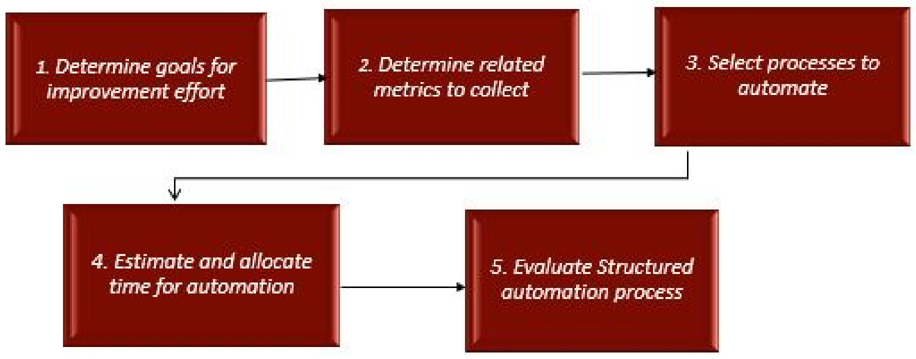

1. Determine goals for improvement effort
2. Determine related metrics to collect
3. Select processes to automate
4. Estimate and allocate time for automation
5. Evaluate Structured automation process

Figure 7.2: DevOps Implementation Path

7.6 Focus Group Suggestion: Model Artefact Repository

Following the discussion on the model, the Focus Group suggested the inclusion of folder structure to demonstrate what is being produced at each step of the model. This suggestion is presented below:

- New project (folder)
  - step zero (folder - requirements before beginning)
    - skill sets file
  - step one (folder)
    - selected strategies files
    - meta model files
    - strategy fragment files
  - step two (folder)
    - situational factors grouped by facets file
    - selected suitable fragments file
    - Skills required vs skill sets file
    - Assembled fragments file
    - Merge strategy fragments
  - step three (folder)
    - Process automation of strategies file
    - Automation plan file
    - Change management plan file

- step four (folder)
  - metrics collection file
  - Impact of change file (if successful)
  - Outcome of change record file (if unsuccessful)
  - Knowledge sharing plan file (if unsuccessful)
  - Devops improvements file (if unsuccessful)

7.7 Practical Instantiation of the DevOps Strategy Model – SIDSEM

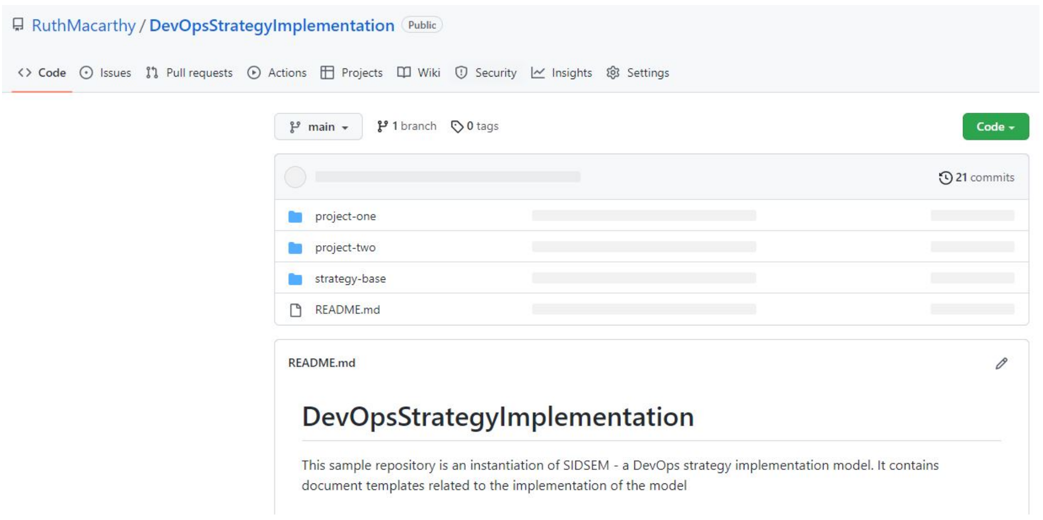
In response to section 7.7, a practical instantiation of the model has been created in a GitHub repository (Macarthy, 2022). This repository has public access, so practitioners can clone and modify its content following the guidelines of the model. The repository further enhances the applicability of the model in these aspects:

1. As GitHub includes version control, practitioners can collaboratively work on improving their practices with a clear view of the entire process, while keeping track of all modifications and changes.
2. The structure of the repository reflects all component parts of the model. As practitioners apply the model, they can keep track of their DevOps journey through a continuous update of the repository for each specified step. This is consistent with the iterative nature of agile practices. The repository provides structured and easily accessed information around an organisations' software development practices.
3. Records from this repository could be used to further study DevOps and the performance of the different DevOps strategies.

The rest of this section describes the repository.

7.7.1 Repository Structure

Figure 7.3 shows an overview of the repository. It consists of two main parts: the strategy base and individual projects. The content of the strategy base is the same for a given organisation. It changes with the method engineers' exposure to DevOps state-of-the-art. The project component is however modified according to the context of individual projects. These are described later.


RuthMacarthy / DevOpsStrategyImplementation Public

project-one
project-two
strategy-base
README.md

README.md

DevOpsStrategyImplementation

This sample repository is an instantiation of SIDSEM - a DevOps strategy implementation model. It contains document templates related to the implementation of the model

Figure 7.3: Repository structure

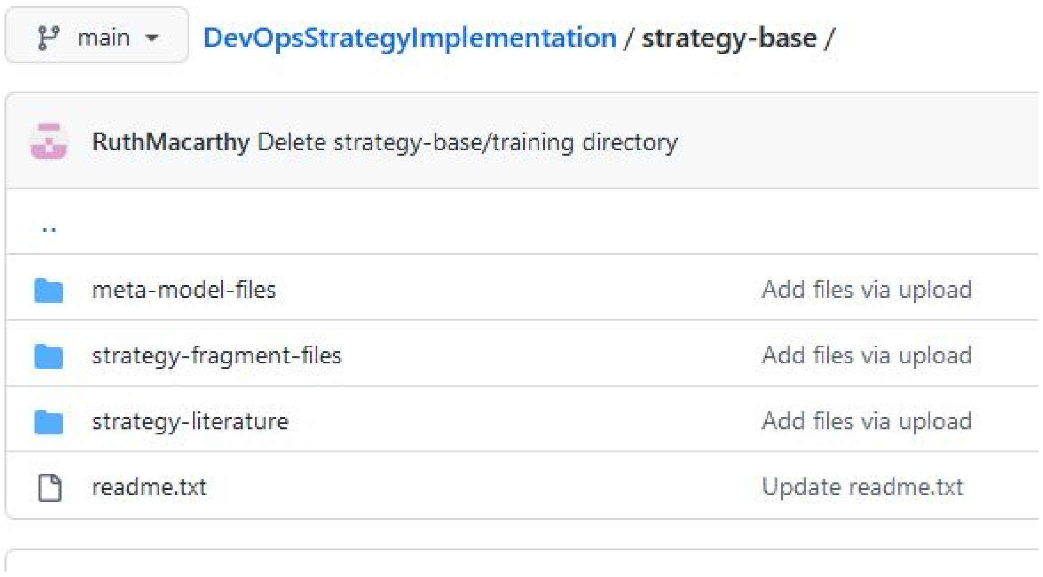
7.7.2 Strategy Base

Figure7.4 depicts the strategy base. It consists of strategy literature, meta-models, and strategy fragments files.

The strategy literature should contain documents relating to the description of DevOps strategies.

The meta-model files describe the meta-model structure used to create strategy fragments.

Created fragments are stored in the strategy fragments files, and accessible by all projects.



main DevOpsStrategyImplementation / strategy-base /

RuthMacarthy Delete strategy-base/training directory

..

meta-model-files	Add files via upload
strategy-fragment-files	Add files via upload
strategy-literature	Add files via upload
readme.txt	Update readme.txt

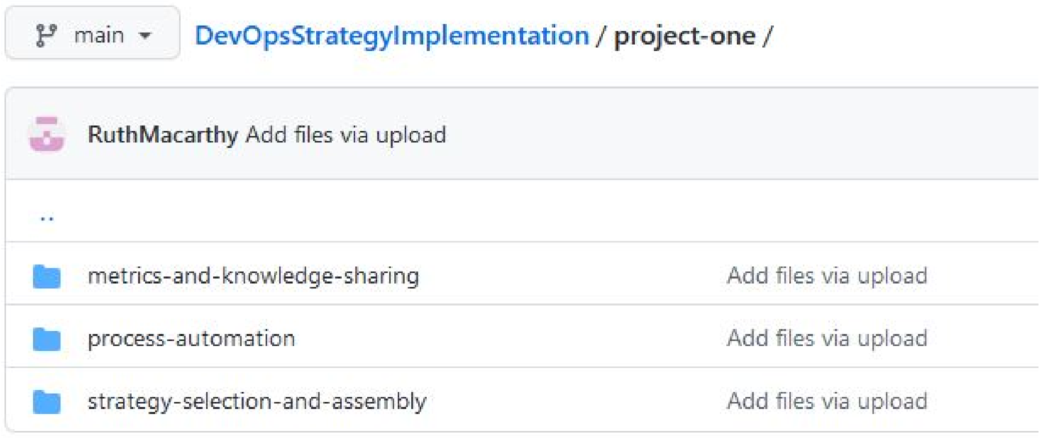
Figure 7.4: Strategy base

7.7.3 Project Structure

Each instance of a project consists of the other three (3) components of the model: strategy selection and assembly, process automation, and metrics and knowledge sharing. Figure 7.5 illustrates this.

7.7.4 Strategy Selection and Assembly

Following the guidelines of SIDSEM, the method engineers (or any practitioner responsible for the process determination in the software development value chain) selects suitable fragments from the strategy base, merge related fragments and assemble them, identify the required skillset and training where applicable. Each of these steps are recorded in the appropriate file as shown in Figure 7.6.



main DevOpsStrategyImplementation / project-one /

RuthMacarthy Add files via upload

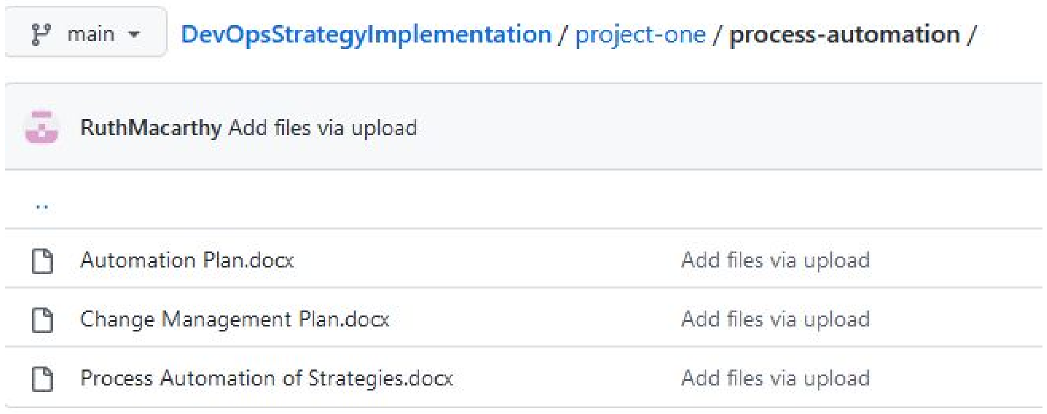
..

metrics-and-knowledge-sharing	Add files via upload
process-automation	Add files via upload
strategy-selection-and-assembly	Add files via upload

Figure 7.5: Project structure

7.7.5 Process Automation

Figure 7.7 shows the process automation documentation. It contains documents related to identified automation candidates, automation plan, and change management plan.



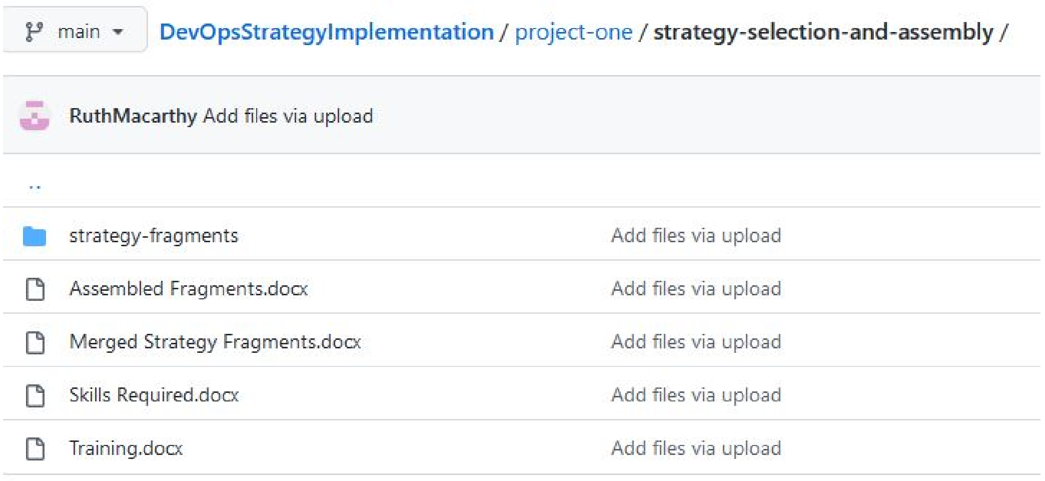
main DevOpsStrategyImplementation / project-one / process-automation /

RuthMacarthy Add files via upload

..

Automation Plan.docx	Add files via upload
Change Management Plan.docx	Add files via upload
Process Automation of Strategies.docx	Add files via upload

Figure 7.7: Process automation



main DevOpsStrategyImplementation / project-one / strategy-selection-and-assembly /

RuthMacarthy Add files via upload

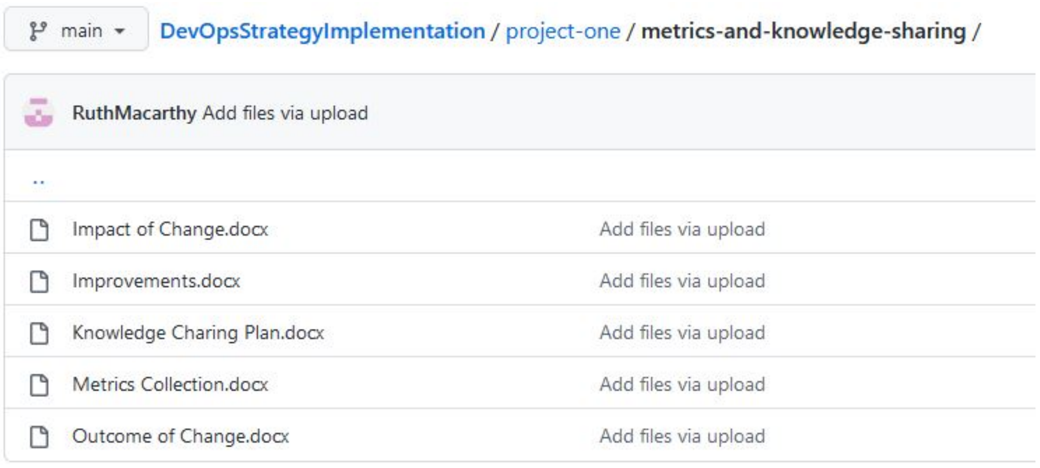
..

strategy-fragments	Add files via upload
Assembled Fragments.docx	Add files via upload
Merged Strategy Fragments.docx	Add files via upload
Skills Required.docx	Add files via upload
Training.docx	Add files via upload

Figure 7.6: Project structure

7.7.6 Metrics Collection and Knowledge-sharing

Lastly, figure 7.8 depicts the metrics collection and knowledge sharing aspect of the model. It contains documents such as collected metrics, outcomes of change, impacts of change, further improvement and knowledge-sharing plans.



main DevOpsStrategyImplementation / project-one / metrics-and-knowledge-sharing /

RuthMacarthy Add files via upload

..

Impact of Change.docx	Add files via upload
Improvements.docx	Add files via upload
Knowledge Charing Plan.docx	Add files via upload
Metrics Collection.docx	Add files via upload
Outcome of Change.docx	Add files via upload

Figure 7.8: Metrics Collection

Chapter 8

Discussion

8.1 Introduction

This chapter examines and interprets research findings in conjunction with related literature, to answer the research questions. The reader is also presented with the contribution of the study to theory and practice. The chapter concludes with identified limitation of the study.

8.2 Answering Research Questions (RQs)

The aim of this research is to propose a model for tailoring DevOps implementation based on organisational context. This goal propelled an in-depth investigation into the patterns of work coordination between development and operations activities in organisations and the elements of influence. To realise the goal, this research answers the two focal questions presented in section 1.3.

8.2.1 RQ1: How do practitioners describe and implement DevOps in practice?

The purpose of this question is to explore the state of the practice of DevOps in comparison with literature. RQ1 was divided into 4 sub-questions to specifically address components parts of the main question. The answer to RQ1 is presented in line with the sub-questions.

8.2.1.1 RQ1a - What are practitioners' perceptions of DevOps definition and description?

A critical examination of the findings shows DevOps being described by practitioners in the study as not just a culture and specific job description, but also distinct teams separate from both developers and operations teams. Although members of these teams have backgrounds in either software development or operations, the nomenclature "DevOps" now separates them from their original silos and classifies them as a unique team of "platform builders" as DvOps1 described it. This seems consistent with the 2014 State of DevOps report of a growing number of DevOps teams (Forsgren et al., 2014). In Humble (2012), it is mentioned that Ops teams may sometimes be referred to as "DevOps team" in some organisations when they build self-service platforms, provide tool chain, training and support developers and the platforms. However, he argues strongly that the insertion of "another layer of indirection between the dev and ops team and call it a 'Devops team'" is not the right way to address the existing gap. Intriguingly, most teams under study fit nicely into both sides of Humble's argument - they are teams between developers and operations teams who provide all the above-mentioned services. This research found no consensus on the definition of DevOps (Erich et al., 2017). In a way, this is not an unexpected finding as most of the practice seem to stem from literature. Participants were observed severally referring to literature during the interviews. There seem to be an agreement on the purpose of DevOps among practitioners as most participants claim that the practice had increased agility in their software development. These claims echo the assertions of other studies (Smeds et al., 2015; Lwakatare et al., 2019). Measurement of such improvement to agility includes delivery speed, recovery from failure time, and deployment frequency etc.

8.2.1.2 RQ1b - How are DevOps functions different from IT Operations and development teams' functions?

To answer the above question, I described the functions in DevOps practices identified in the study. Generally, the responsibilities from development to deployment include creating a piece of software, testing, building and maintenance of automated deployment pipelines, integration, deployment, providing metrics, and platform migration. However, personnel carrying out these functions vary, depending on the DevOps approach adopted by an organisation (the approaches will be discussed in the next section). A summary of the differences is presented in Table 4.2. From the table, a clear distinction in functions is particularly noticed in the Developers-DevOps-Ops mode (Figure 4.5). In this approach, IT Ops teams provide services to DevOps teams in the form of physical infrastructure, DevOps teams provide services to developers in the form of tooling and automated pipelines, while developers provide business value in the form of applications, features etc. The responsibilities of the DevOps teams can simply be described as creating a bridge between previously siloed development and IT operations activities (Nybom et al., 2016).

8.2.1.3 RQ1c: How is DevOps implemented in practice?

A novel taxonomy of four approaches to DevOps, based on interviewees' description of the concept and their practice of it, demonstrates how DevOps is implemented in practice. This is shown in Figure 4.5. As described in Chapter 4, the approaches are developers' interaction with On-premises Ops, Outsourced Ops, DevOps teams, and DevOps bridge teams. These approaches describe the collaboration between developers and DevOps teams based on automation, mixed responsibilities of developers without an IT Ops team, developers working with DevOps teams only, and DevOps teams serving as a bridge between developers and IT Ops teams respectively. The identified approaches are similar to (Nybom et al., 2016) and (López-Fernández et al., 2021). Nybom et al. (2016) suggested three possible approaches of mixed responsibilities, mixed personnel, and bridge team, in which they provided evidence and analysed one of the said approaches (mixed responsibilities). This study however differs from theirs in the following ways: firstly, I provide empirical data confirming practitioners' implementation of all three approaches mentioned in their studies. Secondly, I present a novel mapping of the approaches to on-premises and cloud-based deployments, thereby presenting a fourth approach - which is essentially a variant of mixed responsibilities but can only be found in cloud-based deployment environment. Thirdly, I identified the "facilitators" of DevOps practices in the four approaches. López-Fernández et al. (2021) presented a taxonomy of 4 team structures: i) interdepartmental Dev & Ops collaboration, ii) Interdepartmental Dev-Ops team - the unification of development and operation teams, iii) Boosted cross-functional DevOps team, and iv) Full cross-functional DevOps team. Their characterisation was based on factors such as Leadership from management, Shared product ownership, etc.

Table 4.1 shows Devops bridge team approach to be the most common encountered in the study. While this is not a quantitative study to investigate the frequency of occurrence of each approach, this study assumes that the seeming popularity of the approach might be because DevOps is still evolving, and organisations try to minimise interruptions to existing structures.

8.2.1.4 RQ1d: What strategies are employed in DevOps implementation?

The study identified 6 distinct strategies adopted by organisations to implement DevOps namely, Platform, Greenfield Application, Monolith Decomposition, Process Improvement, Cultural Improvement, and Advocacy. Based on participants description, these strategies seem like subsets of the taxonomy of DevOps approaches in Fig. 4.2 (Dev-Ops, Dev-OutOps The same as Developer-Outsourced

Ops approach, Dev-DevOps, and DevOps bridge teams). They seem to be the initial steps taken by the organisations in their DevOps journey. This is described in the Fig. 8.1. The path that leads from individual strategies to the approaches in Figure 4.2 is not explored in this study. Relating this to the study by Leite et al. (2022), I see links between Greenfield application strategy and single department structure usually adopted by startups, monolith decomposition strategy and API-mediated departments, both of which seek to overcome bottlenecks, and cultural improvement strategy and collaborative department.
The 2020 State of DevOps Report (Brown et al., 2020) provides an in-depth investigation into internal platforms, describing advantages such as improved efficiency of application teams, improved governance, continuous infrastructure

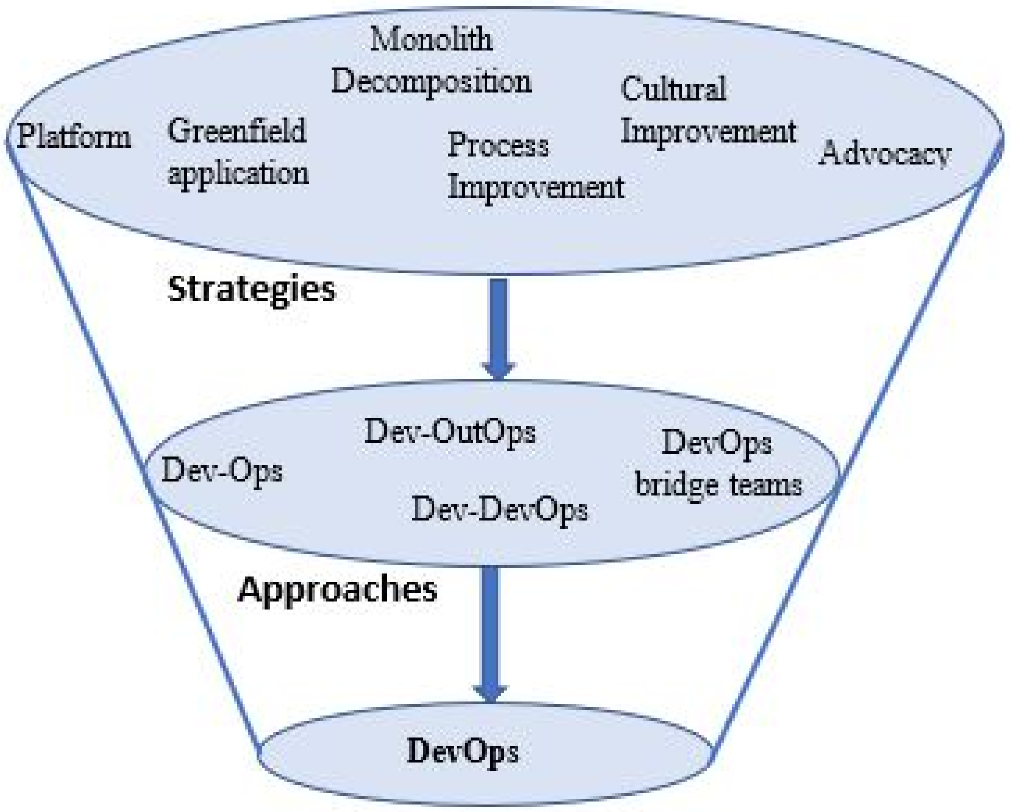


Figure 8.1: DevOps Implementation Path

improvement, and an end to context-switching. It also identifies lack of time, standardisation, and technical skills within the team as challenges. My study differs from this in the sense that I identified and described five more strategies not previously explored in literature, to the best of my knowledge.

Comparing these strategies with the definition of DevOps by Donovan Brown, "DevOps is the union of people, process, and products to enable continuous delivery of value to our end users" (Guckenheimer, 2018), I see the elements of people, process, and technology in various degrees. However, the organisations seem to begin their implementation of DevOps with focus on either one or two of these elements. The Platform and Greenfield application strategies seem to be the most "techno-centric". A lot of emphasis is placed on appropriate tool-set to speed up the delivery and feedback process. There is minimal and often no need for initial restructuring of teams or organisational redesign, and no mention of training at the onset. Reasonable effort is also put into improving the processes in software development. Monolith Decomposition and Process improvement strategies are both "process-centric" and "techno-centric". Effort is put into improving the overall software development process by identifying where the costs are in their work from idea to useful software in the hands of users, improving the quality and speed of feedback, and removing waste from the processes. Some participants also emphasise the importance of automation, CI/CD pipelines, and configuration management. As one interviewee said, "So, I talk about it as basically um, the implementation of lean manufacturing and applying principles of lean manufacturing to helping uh, software value streams achieve agility, uh, stability, efficiency, uh quality, security and satisfaction." [DevCo4_CEO]

Cultural improvement and Advocacy strategies are more "people-centric". The concerns in these strategies are better collaboration between development and operations teams, and consequently better visibility of activities and actions on the route to value delivery. The initial focus here seems to be around organising people, making sure they have the right skills and they're in the right team, and then organising the flow of work in adherence to agile and lean thinking, eliminating waste, and ultimately speeding up the work.

Based on the findings of the study, it can be said that while organisations try to implement DevOps through the strategies described, there seem to be no existing structured way to determine which strategy best fits specific situations. According to Conway's law (Conway, 1968), "organisations which design systems are constrained to produce designs which are copies of the communication structures of these organisations" This study observes that the implication of this differs from one DevOps strategy to another. For instance, in the Greenfield application strategy, such constraints as identified by Conway's law are almost unobserved as its structure is unencumbered with dependencies of the organisation. This leads the reader to the second research question.

8.2.2 Research Question 2: "How can organisations tailor DevOps implementation to suit their organisational context?"

This question explores the possibility of designing an adaptive model for DevOps implementation and is answered within two sub-questions.

8.2.2.1 RQ2a - What elements influence the approach taken by organisations to implement DevOps?

Clearly, from literature(Lwakatare et al., 2019; Amaro et al., 2022) and my findings, DevOps has a lot of intricacies in possibilities. At any point in time, the goals of an organisations must be clearly defined otherwise much effort would be put different directions without meaningful value (Leite et al., 2019). The State of DevOps report insists that some organisations still struggle with DevOps implementation, sometimes leading to failure Brown et al. (2020). Findings of this study that practitioners are mostly concerned with identifying their present technical capabilities and the constraints facing them. Evidently, most decisions made on the adoption of DevOps patterns and practices are predicated on these discoveries. We see a strong connection between the technical capabilities, the constraints, and the approach taken by the organisations to implement. For example, in DevCo3, “People that had no history of the organisation were brought in to influence collaboration which was one of the major constraints. So, just going into an organisation like that, that has a lot of history, old systems, old processes, lots of red tapes to take apart because of the size and age of the organisation, and we really had no choice to use an advocacy model.” [DevCo3_Lead. This agrees with the conclusion drawn by Rodrigues et al. (2020). However, the technical capabilities and constraints alone does not seem to be the only determinants of the approach taken by the organisations to implement DevOps. Two participants mentioned skillset availability evaluation as part of their discovery process. According to DevCo4, the decision of the strategy to employ was largely left to the consultants: “we try to measure that and see where the, given whatever their goals are, you know, where we could make improvements. And after that, it’s mostly just um, you know consultants doing their work, using their experience to recommend solutions.” [DevCo4_CEO]

The literature review previously presented show that many factors influence the approach to software development practices adopted by organisations (Kalus & Kuhrmann, 2013; Bass, 2016; Amaro et al., 2022). This study finds these assertions to be true. Analysis of the findings suggests that skillset is critical in the determination of the DevOps strategy to adopt. The significance of skillset and automation in the six DevOps strategies is extensively described in chapter 6, leading to the development of SIDSEM and answer to the final research question.

8.2.2.2 RQ2b: How can DevOps implementation be tailored to suit on organisational context?

This question was predicated on the gap identified by Leite et al. (2019) and Lwakatare et al. (2019). The studies both insisted on the need for a more structured DevOps implementation in organisations. An examination of the research findings culminated in the development of a novel model for DevOps implementation (described in chapter 6). The theories generated from the study necessitated the extension of Brinkkemper (1996) structure of Situational Method Engineering. While Brinkkemper's model suited the tailoring agile software development methods, it does not extend to IT operations process management nor address the influencing factors of DevOps implementation identified in the study. This necessitated a modification of the accepted Brinkkemper's model, which recognises the key role of skillset in the determination of DevOps strategies, and the importance of structured automation and continuous measurement.

Organisational context is well-researched, especially with regards to agile software development. This study describes situational factors summing up to organisational context in chapter 2. To utilise the model, organisations should clearly understand the individual strategies identified in the study. This essentially is the responsibility of the method engineer, who should use the meta models described in chapter 6 to break down each of the strategies to usable fragments. This provides limitless ways to tailor DevOps implementation that suits the context of the organisation.

Clarity and understandability are essential for the acceptance of a model. Findings from the focus group workshop demonstrates practitioners seeming acceptance of the model. Although the complete model seemed complicated to the practitioners at first, they related to the simplified version of the model which helped their understanding of the complete one. The practitioners also suggested the inclusion of a file structure for the model. This is an indication of their understanding of its structure. The model consists of 4 basic parts: 1. Strategy administration and strategy base 2. Selection and assembly of strategy fragments 3. Process automation 4. Metrics and continuous measurement Parts 1 and 2 are similar to Brinkkemper (1996) and Henderson-Sellers et al. (2014) models. In principle, these parts correspond in functionality with the above-metioned models. This research findings show clearly the active roles of process automation and continuous measurement in achieving the goals of DevOps. The step-by-step description of the model provided in chapter 6 increases the usability of the model. The applicability of the model is demonstrated in the Focus group using it to identify the gaps in their DevOps implementation and possible actions to further improve their agility. This lends credence to the suitability of the model to review and improve software development processes. In response to the suggestion of a file structure for the model, a repository was created (Macarthy, 2022). This provides a practical means to document DevOps strategy implementations and keep track of improvements and other changes.

SIDSEM is thus recommended as guidance for organisations who wish to implement DevOps or assess their software development processes, with the goal of improving their agility.

8.3 Contribution to Theory

In line with the interrelated theory structure proposed by Gregor (2006), this thesis contributes to DevOps body of knowledge through the instrumentality of the Theory of Analysis, the Theory of Explaining, and the Theory of Design and Action. The theory of Analysis "describe or classify specific dimensions or characteristics of individuals, groups, situations, or events by summarising the commonalities found in discrete observation" (Fawcett, 1978). This thesis presents a theory of analysis by proposing a taxonomy DevOps approaches taken by organisations. The classification maps the approaches to cloud and on-premises deployments, as well as the facilitators of the practices. The theory of Explaining clarifies "primarily how and why some phenomena occur" (Gregor, 2006). This thesis provides illumination on the strategies organisations employ to implement DevOps. It also examines the the influence of skillet in the determination of a strategy and the role of automation in the strategies. Results indicate exigent correlation between these elements and the identified strategies. The Theory of Design and Action gives prescription for the construction of a process or product. This thesis demonstrates the Theory of Design and action through the development and validation of a model seeking to provide guidance for DevOps implementation in organisations. which is predicated on.

This research displays novel findings centred around the existing nature of DevOps implementation in the software industry, and prescribed steps to improve agility in service delivery with the instrumentality of the proposed model. The results and outcomes presented in this study are contributory to filling the research gaps in DevOps implementation. To the best of the researcher's knowledge, based on extensive search of existing literature, no previous work presents findings as have been provided by this research. The creation of a repository for implementation and documentation of the model usage presents an opportunity for future researchers to collect and further investigate the concept.

8.4 Contribution to Practice

The DevOps implementation model has implications for practitioners. This section presents the implications of the model for management, methods engineers, and software development practitioners (developers, IT operations, DevOps engineers, SREs, Security, QA etc).

For the management of organisations, this study aids clearer understanding of the concept of DevOps. SIDSEM throws more light on whether an organisation should be restructured to adopt DevOps (Leite et al., 2019). As organisations seek the least disruption to their processes during such transformations, the model would help them decide which strategy is most suitable to adopt based on their organisational context. The model also presents an upfront view of the requirements and expectations of the chosen strategy, further improving their decision process. As such, this model could as a tool for decision makers to carve a path for their software delivery, such as deciding to invest in relevant training or technologies.

This research provides an in-depth analysis of the different DevOps strategies to Methods engineers. It describes the activities, requirements, and other constituents of the identified strategies. Method engineers can thus break down the strategies to component fragments and piece them together using the SIDSEM.

Software development practitioners can benefit from the taxonomy of DevOps approaches presented in the study. The taxonomy clearly identifies the roles and responsibilities in the different approaches. SIDSEM provides a structured approach to automation for improved agility. In the absence of methods engineers, the model would help steer the teams towards suitable tailored strategies for their DevOps implementation.

The creation of a repository (Macarthy, 2022), which serves as a physical representation of the model further improves its usability in practice. Practitioners can implement suitable strategies of DevOps using the model, and document the entire process using the repository.

8.5 Limitations of the Study

The study adopts the assessment for rigor and trustworthiness in qualitative research as proposed by Guba & Lincoln (1994): confirmability, dependability, internal consistency (credibility), and transferability. I believe I have done my best to avoid researcher's bias by closely following the research methodology as prescribed by classical Grounded Theory. We describe how this was achieved in the following subsections.

8.5.1 Confirmability

Confirmability denotes the researcher's ability to present a chain of evidence showing that the data is a true representation of participants' responses, and not the researcher's bias or opinion(Adolph et al., 2011; Cope, 2014; Krefting, 1991). Data was collected through semi-structured interview and transcribed by hand. The study follows the prescribed guidelines of Grounded Theory to analyse collected data, from coding to the generation of theory. To ensure confirmability, I described how conclusions and interpretations in the study were established based on codes from transcripts and memos written from related codes. All findings are based on participants ‘quotes.

8.5.2 Dependability

Dependability refers to the repeatability of the study (Adolph et al., 2011; Cope, 2014; Krefting, 1991).The exact methods of data collection, data analysis and interpretation are clearly described in this thesis. A sample of interview questions can be seen in Bass & Macarthy (2020). The step-by-step description makes it possible for another researcher to carry out the same study.

8.5.3 Internal Consistency

Internal consistency indicates credibility and consistency of finding (Adolph et al., 2011; Cope, 2014; Krefting, 1991). To ensure credibility, interviews were transcribed verbatim by hand. Also, participants’ responses are clearly described in this thesis. Appendix E is a transcript from an interview in the study. For consistency, I compared the codes across the organisations in the study till concrete theories were formed.

8.5.4 Transferability

Transferability describes the applicability of the study to other settings (Cope, 2014; Krefting, 1991), and useful theories can be derived from findings (Adolph et al., 2011). Although this study investigates DevOps implementation in 29 organisation and a model was developed based on the finding, I do not claim extendibility to all organisational contexts as the model has not been tested beyond 1 organisation. I however believe that my findings are relevant in decision making for DevOps implementation.

Chapter 9

Conclusion

9.1 Summary of Thesis

The software development process is characterised by interconnections of several activities carried out by different actors and teams. As the demand for more agility in software service delivery continue to rise, so does the need for coordination and collaboration among the stakeholders. Recent events like the Covid-19 pandemic, which has necessitated working from home, has placed more demand on better coordination. An extensive literature review of coordination in software development is presented in this thesis, exploring the efforts of software development methodologies from waterfall to agile methods. These methodologies were however targeted at the development phase alone. Recent contributions to coordination extend the goal of agility to IT operations and the other teams in the delivery value chain, most popular of which is DevOps.

DevOps is described as a cultural movement in software engineering aimed at building, testing and releasing software faster and more reliably through automation. According to available literature, benefits derived from DevOps adoption include faster delivery, improved quality and security, and better collaboration. The 2020

State of DevOps report shows a rising trend of its adoption among practitioners and organisations. Despite its increasing popularity, there are differing descriptions of the concept, resulting in various dissimilar ways of implementation. While some studies characterise it as an organisation-wide culture to foster better collaboration between IT Ops and development teams, DevOps has also been portrayed as a job description by others. Furthermore, current studies identify the lack of guidelines for organisations to successfully implement DevOps. The random ways of implementation and tailoring of practices in the name of DevOps is considered risky as it might lead to significant financial losses and upheaval of the organisation's software development process. The aim of this research is to investigate the patterns of work coordination between development and operations activities in organisations and propose a model for DevOps implementation. To achieve the aim, two main research questions emerged from Grounded Theory process: Research Question 1 -"How do practitioners describe and implement DevOps in practice?" Research Question 2 - "How can organisations tailor DevOps implementation to suit organisational context?" These questions were further divided into sub questions to enable a more detailed investigation.

This study adopts a multi-method approach. Chapter 3 -Research design clearly explains the rationale and philosophical choices of the study. The research is divided into four phases, each of which is extensively described along with the applied methodology. To understand its implementation in practice, phase 1 presents an exploratory study based on interviews with 11 DevOps practitioners across nine organisations. Data was analysed by a method informed by Grounded Theory. The empirical findings in this phase provide new understanding of DevOps implementation, based on the descriptions by the practitioners in the study.

Firstly, an in-depth analysis of the data resulted in the development of a taxonomy of DevOps implementation in practice. Four modes of implementation are identified, categorised as developers' interactions with: Ops, Outsourced Ops, DevOps teams, and DevOps bridge teams. Secondly, a novel mapping of these approaches to cloud and on-premises deployment environments is presented (Macarthy & Bass, 2020). Thirdly, the facilitators of DevOps practices in these modes are identified. Finally, further analysis of the fourth approach exposed three distinct groups of activities: provisioning and maintenance of physical systems, function virtualisation and creation of automated pipelines, and development, deployment and maintenance of applications. This, I believe, may have given rise to the implementation of DevOps as teams between developers and operations teams, as each group of activities require specific skillsets. The findings revealed that DevOps is indeed perceived by participants as both a culture and a job description. However, the two views did not seem mutually exclusive.

Phase 2 is an in-depth investigation of DevOps implementation based on interviews with DevOps practitioners from 18 organisations, based on findings from phase 1. This followed the same analysis method as phase 1. The findings include six strategies identified to be used by organisations for DevOps implementation: platform, greenfield application, monolith decomposition, process improvement, cultural improvement, and advocacy (Macarthy & Bass, 2021). Except for the platform strategy, all the others are uniquely characterised by this research. The roles of skillet and automation in each strategy was investigated. The actual relationship between the skillset and DevOps strategy has not been established. However, there are indications from my findings that the skillset impacts on the identified strategies in areas such as the duration and overall outcome of the implementation effort. This leads us to the conclusion that a striking correlation exists between the skillset and the strategy adopted by organisations to implement DevOps.

The findings from these phases, and the theories generated, combined with literature led to the development of a model for DevOps implementation in phase 3. This model is a extension of Brinkkemper (1996) situational method engineering model, to address the identified requirements of DevOps. The model was validated in phase 4 using an expert focus group. The focus group evaluated their software development process with the model and identified ways in which they could further improve their agility.

The answers to the research questions are discussed in chapter 8, reflecting the findings from the 4 phases of the study. The main contributions of this thesis are: a novel taxonomy of DevOps approaches in practice, the identification and characterisation of 6 DevOps strategies, and the development of a novel model for DevOps strategy implementation.This study also provides the instantiation of the model in a GitHub repository, providing practitioners with a practical means of applying the model and documenting their DevOps journey.

This thesis concludes that amid the ambivalence about the definition and practices of DevOps resulting in the unstructured implementation of the concept, the model developed in the study can guide organisations to make informed decisions on the most suitable DevOps strategy to employ according to their organisational context.

9.2 Reflection

The PhD journey is mostly about how one has developed as a researcher. Deciding on a PhD was not as easy choice as I already had an industry-based career in computer networks. I also had a family to think about. The corporation I worked with decided to train researchers for the R&D division and I was chosen for the highly competitive Petroleum Technology Development Fund (PTDF) scholarship. As hard as it was, I could not pass up the opportunity to make a contribution to the field of information systems through a PhD.

It is highly unlikely to predict how things will go at the start of the PhD but certainly, changing circumstances will impact you both academically and personally. In this section, I reflect on the main stages of this study from choosing the goal, deciding on the methodology of inquiry, collecting and analysing data, and making contributions.

The area of inquiry I chose is still developing. Deciding on the aim and objectives, and crafting the research questions was not a straightforward task. This was mainly because of the methodology I chose to kick-off the investigation - Grounded Theory. The research questions developed along with the study. Understanding the different methodologies and the role they play in research is key for anyone who wishes to embark on a PhD study. It is also important to ensure your supervisor is conversant with your chosen methodology. Guidance in the application of a methodology is indispensable if one must develop their knowledge on conducting research.

I collected data in two iterations and both times, it was challenging for various reasons. In the first instance, I was new to the UK and did not know how to reach software development practitioners. Thanks to my supervisor who willingly introduced me suitable industry associates, I started with purposive sampling then carried on with snowballing method. The second phase of data collection was at the peak of Covid-19. Physical offices were closed up, organsations were restructuring, and it was almost impossible to speak with practitioners, most of whom were struggling with personal impacts of the pandemic. I could not collect data for more than 5 months. I resorted to seeking participants on LinkedIn. This involved determining their suitability through a painstaking examination of their profiles. I daringly contacted industry leaders in the DevOps movement through LinkedIn. Thankfully, I got some responses. Again, I employed snowballing method and collected a substantial amount of data.

Applying the guidelines Grounded Theory in the data analysis was difficult in my first iteration. I needed to understand how to code the transcripts both by hand and using Nvivo. By the second iteration, I could not recommend the methodology enough for exploratory investigations. It was rewarding to see theories emerging from the process.

Writing up is a stage to pay close attention to. I would say this process is incremental and starts at the beginning of the PhD. It builds up from the literature review and other written works in the first year, to publications achieved during the study. Some might not be useful for the final version of the thesis but one would not start from scratch.

In the course of this journey, I had several family losses, my father-in-law, my grandmother, my aunt to a ghastly motor accident, and a dear friend cut down in her prime. All of these impacted my studies significantly. I had other personal struggles to deal with too, not to mention the upheaval caused by the pandemic. I am thankful that I am able to complete this study.

9.3 Future Work

DevOps is still an emerging concept and its popularity continues to grow. More research is required to further explore the nature of the concept and arrive at a formal definition. This study did not explore the relationship between the identified strategies and the approaches to DevOps implementation. An investigation might be relevant to provide organisations with information on possible team topologies that might arise from the strategies they adopt. Further research on this topic needs to be undertaken before the association between the skillset and strategies in DevOps implementation is more clearly understood. The model developed in this study is evaluated using an expert focus group workshop. Further validation of the model needs to be carried out, to establish its suitability as guidance for DevOps implementation. The next step with this research would be recruiting organisations to use the model together with the repository.

References

Abrahamsson, P., Conboy, K., & Wang, X. (2009). *'lots done, more to do': the current state of agile systems development research* (Vol. 18) (No. 4). Taylor & Francis.

Adams, W. C. (2015). Conducting semi-structured interviews. *Handbook of practical program evaluation*, *4*, 492–505.

Adolph, S., Hall, W., & Kruchten, P. (2011). Using grounded theory to study the experience of software development. *Empirical Software Engineering*, *16*(4), 487–513.

Ali, N. B., Petersen, K., & Schneider, K. (2016). Flow-assisted value stream mapping in the early phases of large-scale software development. *Journal of Systems and Software*, *111*, 213–227.

Amaro, R. M. D., Pereira, R., & da Silva, M. M. (2022). Capabilities and practices in devops: A multivocal literature review. *IEEE Transactions on Software Engineering*.

Anderson, D. J. (2010). *Kanban: successful evolutionary change for your technology business*. Blue Hole Press.

Andres, H. P., & Zmud, R. W. (2002). A contingency approach to software project coordination. *Journal of Management Information Systems*, *18*(3), 41–70.

Avison, D. (1996). Information systems development methodologies: a broader perspective. In *Working conference on method engineering* (pp. 263–277).

Aydin, M. N. (2007). Examining key notions for method adaptation. In *Working conference on method engineering* (pp. 49–63).

Baddoo, N., & Hall, T. (2003). De-motivators for software process improvement: an analysis of practitioners' views. *Journal of Systems and Software*, *66*(1), 23–33.

Bass, J. M. (2014). Scrum master activities: process tailoring in large enterprise projects. In *2014 ieee 9th international conference on global software engineering* (pp. 6–15).

Bass, J. M. (2016). Artefacts and agile method tailoring in large-scale offshore software development programmes. *Information and Software Technology*, *75*.

Bass, J. M., & Haxby, A. (2019). Tailoring product ownership in large-scale agile projects: managing scale, distance, and governance. *IEEE Software*, *36*(2), 58–63.

Bass, J. M., & Macarthy, R. W. (2020). *Interview guide - devops (version 1).* University of Salford. Retrieved from https://doi.org/10.17866/rd.salford.12349724.v1

Baumoel, U. (2005). Strategic agility through situational method construction. In *Proceedings of the european academy of management annual conference* (Vol. 2005).

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., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., & Thomas, D. (2001). Manifesto for agile software development twelve principles of agile software. *Zugriff am*, *5*, 2020.

Beecham, S., Noll, J., Richardson, I., & Ali, N. (2010). Crafting a global teaming model for architectural knowledge. In *2010 5th ieee international conference on global software engineering* (pp. 55–63).

Beecham, S., OLeary, P., Richardson, I., Baker, S., & Noll, J. (2013). Who are we doing global software engineering research for? In *2013 ieee 8th international conference on global software engineering* (pp. 41–50).

Bergstra, J., Jonkers, H., & Obbink, J. (1985). A software development model for method engineering. *Esprit*, *84*, 84–94.

Boehm, B., & Huang, L. G. (2003). Value-based software engineering: a case study. *Computer*, *36*(3), 33–41.

Boehm, B., & Turner, R. (2004). Balancing agility and discipline: Evaluating and integrating agile and plan-driven methods. In *Proceedings. 26th international conference on software engineering* (pp. 718–719).

Boote, D. N., & Beile, P. (2005). Scholars before researchers: On the centrality of the dissertation literature review in research preparation. *Educational researcher*, *34*(6), 3–15.

Brinkkemper, S. (1996). Method engineering: engineering of information systems development methods and tools. *Information and software technology*, *38*(4), 275–280.

Brinkkemper, S., Saeki, M., & Harmsen, F. (1998). Assembly techniques for method engineering. In *International conference on advanced information systems engineering* (pp. 381–400).

Brown, A., Kersten, N., & Stahnke, M. (2020). State of devops report 2020. *Available from: puppet. com/resources/report/state-of-devopsreport*, *16*.

Bucher, T., Klesse, M., Kurpjuweit, S., & Winter, R. (2007). Situational method engineering. In *Working conference on method engineering* (pp. 33–48).

Cameron, K. S., & Quinn, R. E. (2011). *Diagnosing and changing organizational culture: Based on the competing values framework*. John Wiley & Sons.

Cao, L., Mohan, K., Xu, P., & Ramesh, B. (2009). A framework for adapting agile development methodologies. *European Journal of Information Systems*, *18*(4), 332–343.

Charmaz, K. (2009). Shifting the grounds. *Developing grounded theory: The second generation*, 127–154.

Chen, L. (2015). Continuous delivery: Huge benefits, but challenges too. *IEEE software*, *32*(2), 50–54.

Chen, L. (2017). Continuous delivery: Overcoming adoption challenges. *Journal of Systems and Software*, *128*, 72–86.

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.

Clark, J. S., Porath, S., Thiele, J., & Jobe, M. (2020). *Action research.* New Prairie Press.

Clarke, P., & O'Connor, R. V. (2012). The situational factors that affect the software development process: Towards a comprehensive reference framework. *Information and software technology*, *54*(5), 433–447.

Cockburn, A. (2004). *Crystal clear: A human-powered methodology for small teams: A human-powered methodology for small teams.* Pearson Education.

Conway, M. E. (1968). How do committees invent. *Datamation*, *14*(4), 28–31.

Cope, D. G. (2014). Methods and meanings: credibility and trustworthiness of qualitative research. In *Oncology nursing forum* (Vol. 41).

Creswell, J. W., & Creswell, J. D. (1994). *Research design.* Thousand Oaks, CA: Sage.

Creswell, J. W., & Creswell, J. D. (2017). *Research design: Qualitative, quantitative, and mixed methods approaches.* Sage publications.

Dearle, A. (2007). Software deployment, past, present and future. In *Future of software engineering (fose'07)* (pp. 269–284).

Debois, P. (2008). Agile infrastructure and operations: how infra-gile are you? In *Agile 2008 conference* (pp. 202–207).

Dennehy, D., & Conboy, K. (2017). Going with the flow: An activity theory analysis of flow techniques in software development. *Journal of Systems and Software*, *133*, 160–173.

Díaz, J., Almaraz, R., Pérez, J., & Garbajosa, J. (2018). Devops in practice: an exploratory case study. In *Proceedings of the 19th international conference on agile software development: Companion* (pp. 1–3).

Dietrich, P., Kujala, J., & Artto, K. (2013). Inter-team coordination patterns and outcomes in multi-team projects. *Project Management Journal*, *44*(6), 6–19.

Dingsøyr, T., & Lassenius, C. (2016). Emerging themes in agile software development: Introduction to the special section on continuous value delivery. *Information and Software Technology*, *77*, 56–60.

Dingsøyr, T., Nerur, S., Balijepally, V., & Moe, N. B. (2012). *A decade of agile methodologies: Towards explaining agile software development* (Vol. 85) (No. 6). Elsevier.

Dingsøyr, T., Rolland, K., Moe, N. B., & Seim, E. A. (2017). Coordination in multi-team programmes: An investigation of the group mode in large-scale agile software development. *Procedia computer science*, *121*, 123–128.

Dove, R., & Turkington, G. (2008). Relating agile development to agile operations. In *Conference on systems engineering research (cser), university of southern california, redondo beach, april, www. parshift. com/files/psidocs/-pap080404cser2008devopsmigration. pdf.*

Duvall, P. M., Matyas, S., & Glover, A. (2007). *Continuous integration: improving software quality and reducing risk.* Pearson Education.

Dybå, T., & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and software technology*, *50*(9-10), 833–859.

Dyck, A., Penners, R., & Lichter, H. (2015). Towards definitions for release engineering and devops. In *2015 ieee/acm 3rd international workshop on release engineering* (pp. 3–3).

Engström, E., & Runeson, P. (2010). A qualitative survey of regression testing practices. In *International conference on product focused software process improvement* (pp. 3–16).

Erich, F. M., Amrit, C., & Daneva, M. (2017). A qualitative study of devops usage in practice. *Journal of Software: Evolution and Process*, *29*(6), e1885.

Erickson, J., Lyytinen, K., & Siau, K. (2005). Agile modeling, agile software development, and extreme programming: the state of research. *Journal of Database Management (JDM)*, *16*(4), 88–100.

Essink, L. J. (1986). A modelling approach to information system development. In *Proc. of the ifip wg 8.1 working conference on information systems design methodologies: improving the practice* (pp. 55–86).

Etikan, I., Musa, S. A., & Alkassim, R. S. (2016). Comparison of convenience sampling and purposive sampling. *American journal of theoretical and applied statistics*, *5*(1), 1–4.

Fawcett, J. (1978). The relationship between theory and research: A double helix. *Advances in Nursing Science*, *1*(1), 49–62.

Firesmith, D. G., & Henderson-Sellers, B. (2002). *The open process framework: An introduction*. Pearson Education.

Fitzgerald, B., Hartnett, G., & Conboy, K. (2006). Customising agile methods to software practices at intel shannon. *European Journal of Information Systems*, *15*(2), 200–213.

Forsgren, N., Kim, G., & Labs, P. (2014). *2014 state of devops report*. Retrieved from http://puppetlabs.com/sites/default/files/2014-state-of-devops-report.pdf

Fowler, M., & Foemmel, M. (2006). Continuous integration. thought-works. *Inc.[Online: accessed in May 16, 2018 from http://www. martinfowler. com/.... html].*

Fruhling, A. L., & Tarrell, A. E. (2008). Best practices for implementing agile methods. *IBM center for the Business government*.

Galbraith, J. (1973). Designing complex organizations. *Reading, Mass*.

Gill, A. Q., Loumish, A., Riyat, I., & Han, S. (2018a). Devops for information management systems. *VINE Journal of Information and Knowledge Management Systems*.

Gill, A. Q., Loumish, A., Riyat, I., & Han, S. (2018b). Devops for information management systems. *VINE Journal of Information and Knowledge Management Systems*.

Ginsberg, M. P., & Quinn, L. H. (1995). Process tailoring and the the software capability maturity model. *SEI Joint Program Technical Report for the U.S. Department of Defense*.

Giray, G., & Tekinerdogan, B. (2018). Situational method engineering for constructing internet of things development methods. In *International symposium on business modeling and software design* (pp. 221–239).

Glaser, B. G. (1978). Theoretical sensitivity. *Advances in the methodology of grounded theory*.

Glaser, B. G. (1992). *Basics of grounded theory analysis: Emergence vs. forcing. 1992, mill valley.* CA: Sociology Press.

Glaser, B. G. (1998). *Doing grounded theory: Issues and discussions*. Sociology Press.

Glaser, B. G., & Strauss, A. (1967). Grounded theory: The discovery of grounded theory. *Sociology the journal of the British sociological association*, *12*(1).

Glass, R. L. (2003). Questioning the software engineering unquestionables. *IEEE Software*, *20*(3), 120.

Gonzalez-Perez, C. (2007). Supporting situational method engineering with iso/iec 24744 and the work product pool approach. In *Working conference on method engineering* (pp. 7–18).

Gonzalez-Perez, C., & Henderson-Sellers, B. (2005). Templates and resources in software development methodologies. *Journal of Object Technology*.

Goodman, D., & Elbaz, M. (2008). " it's not the pants, it's the people in the pants" learnings from the gap agile transformation what worked, how we did it, and what still puzzles us. In *Agile 2008 conference* (pp. 112–115).

Gregor, S. (2006). The nature of theory in information systems. *MIS quarterly*, 611–642.

Guba, E. G., & Lincoln, Y. S. (1994). Competing paradigms in qualitative research. *Handbook of qualitative research*, *2*(163-194), 105.

Guckenheimer, S. (2018). *What is devops?* Retrieved from https://docs.microsoft.com/en-us/azure/devops/learn/what-is-devops

Gupta, M., George, J. F., & Xia, W. (2019). Relationships between it department culture and agile software development practices: An empirical investigation. *International Journal of Information Management*, *44*, 13–24.

Gutierrez, G., Garzas, J., de Lena, M. T. G., & Moguerza, J. M. (2018). Self-managing: An empirical study of the practice in agile teams. *IEEE Software*, *36*(1), 23–27.

Haghighatkhah, A., Mäntylä, M., Oivo, M., & Kuvaja, P. (2018). Test prioritization in continuous integration environments. *Journal of Systems and Software*, *146*, 80–98.

Hall, R. S., Heimbigner, D., & Wolf, A. L. (1999). A cooperative approach to support software deployment using the software dock. In *Proceedings of the 21st international conference on software engineering* (pp. 174–183).

Harmsen, A. F., Brinkkemper, J. N., & Oei, J. H. (1994). *Situational method engineering for information system project approaches*. University of Twente, Department of Computer Science.

Harmsen, A. F., Ernst, M., & Twente, U. (1997). *Situational method engineering*. Citeseer.

Hawryszkiewycz, I. (2002). Designing collaborative business systems. In *Ifip world computer congress, tc 8* (pp. 131–145).

Heath, H., & Cowley, S. (2004). Developing a grounded theory approach: a comparison of glaser and strauss. *International journal of nursing studies*, *41*(2), 141–150.

Henderson-Sellers, B., & Gonzalez-Perez, C. (2005). The rationale of powertype-based metamodelling to underpin software development methodologies. In *Conferences in research and practice in information technology series.*

Henderson-Sellers, B., Ralyté, J., Ågerfalk, P. J., & Rossi, M. (2014). *Situational method engineering*. Springer.

Highsmith, J., & Cockburn, A. (2001). Agile software development: The business of innovation. *Computer*, *34*(9), 120–127.

Highsmith, J. A., & Highsmith, J. (2002). *Agile software development ecosystems*. Addison-Wesley Professional.

Hoda, R., Noble, J., & Marshall, S. (2010). Using grounded theory to study the human aspects of software engineering. *Human Aspects of Software Engineering*, 1–2.

Hoda, R., Noble, J., & Marshall, S. (2012). Developing a grounded theory to explain the practices of self-organizing agile teams. *Empirical Software Engineering*, *17*(6), 609–639.

Hoegl, M., & Weinkauf, K. (2005). Managing task interdependencies in multi-team projects: A longitudinal study. *Journal of Management Studies*, *42*(6), 1287–1308.

Hoppenbrouwers, S., Zoet, M., Versendaal, J., & Weerd, I. v. d. (2011). Agile service development: a rule-based method engineering approach. In *Working conference on method engineering* (pp. 184–189).

Humble, J. (2012). *There's no such thing as a devops team.* Retrieved from https://continuousdelivery.com/2012/10/theres-no-such-thing-as-a-devops-team/

Humble, J., & Farley, D. (2010). *Continuous delivery: reliable software releases through build, test, and deployment automation.* 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.

Hüttermann, M. (2012). *Devops for developers.* Apress.

ISO. (2014). Software engineering - metamodel for development methodologies. *International Standardization Organization, Geneva.*

Iyawa, G. E., Herselman, M. E., & Coleman, A. (2016). Customer interaction in software development: A comparison of software methodologies deployed in namibian software firms. *The Electronic Journal of Information Systems in Developing Countries*, *77*(1), 1–13.

Kalus, G., & Kuhrmann, M. (2013). Criteria for software process tailoring: a systematic review. In *Proceedings of the 2013 international conference on software and system process* (pp. 171–180).

Karlsson, F., & Ågerfalk, P. J. (2004). Method configuration: adapting to situational characteristics while creating reusable assets. *Information and Software Technology*, *46*(9), 619–633.

Kersten, M. (2018a). A cambrian explosion of devops tools. *IEEE Computer Architecture Letters*, *35*(02), 14–17.

Kersten, M. (2018b). Mining the ground truth of enterprise toolchains. *IEEE Software*(3), 12–17. doi: https://doi.org/10.1109/MS.2018.2141029

Kersten, M. (2018c). What flows through a software value stream? *IEEE Software*, *35*(4), 8–11.

Kerzner, H. (1998). *In search of excellence in project management*. Van Nostrand Reinhold.

Kim, G., Humble, J., Debois, P., & Willis, J. (2016). *The devops handbook:: How to create world-class agility, reliability, and security in technology organizations*. IT Revolution.

Kornyshova, E., Deneckère, R., & Salinesi, C. (2007). Method chunks selection by multicriteria techniques: an extension of the assembly-based approach. In *Working conference on method engineering* (pp. 64–78).

Kraut, R. E., & Streeter, L. A. (1995). Coordination in software development. *Communications of the ACM*, *38*(3), 69–82.

Krefting, L. (1991). Rigor in qualitative research: The assessment of trustworthiness. *American journal of occupational therapy*, *45*(3), 214–222.

Kromhout, B. (2017). Containers will not fix your broken culture (and other hard truths). *Queue*, *15*(6), 46–56.

Kruchten, P. B. (1995). The 4+ 1 view model of architecture. *IEEE software*, *12*(6), 42–50.

Kumar, K., & Welke, R. J. (1992). Methodology engineeringr: a proposal for situation-specific methodology construction. In *Challenges and strategies for research in systems development* (pp. 257–269).

Lei, H., Ganjeizadeh, F., Jayachandran, P. K., & Ozcan, P. (2017). A statistical analysis of the effects of scrum and kanban on software development projects. *Robotics and Computer-Integrated Manufacturing*, *43*, 59–67.

Leidner, D. E., & Kayworth, T. (2006). A review of culture in information systems research: Toward a theory of information technology culture conflict. *MIS quarterly*, 357–399.

Leite, L., Lago, N., Melo, C., Kon, F., & Meirelles, P. (2022). A theory of organizational structures for development and infrastructure professionals. *IEEE Transactions on Software Engineering*.

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.

Len, B., Jeffery, R., Wada, H., Weber, I., & Zhu, L. (2013). Eliciting operations requirements for applications. In *Proceedings of the 1st international workshop on release engineering* (pp. 5–8).

Len, B., Weber, I., & Zhu, L. (2015). *Devops: A software architect's perspective*. Addison-Wesley Professional.

Leppänen, M., Mäkinen, S., Pagels, M., Eloranta, V.-P., Itkonen, J., Mäntylä, M. V., & Männistö, T. (2015). The highways and country roads to continuous deployment. *Ieee software*, *32*(2), 64–72.

López-Fernández, D., Diaz, J., Garcia-Martin, J., Pérez, J., & Gonzalez-Prieto, A. (2021). Devops team structures: Characterization and implications. *IEEE Transactions on Software Engineering*.

Luz, W. P., Pinto, G., & Bonifácio, R. (2018). Building a collaborative culture: a grounded theory of well succeeded devops adoption in practice. In *Proceedings of the 12th acm/ieee international symposium on empirical software engineering and measurement* (pp. 1–10).

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.

Lwakatare, L. E., Karvonen, T., Sauvola, T., Kuvaja, P., Olsson, H. H., Bosch, J., & Oivo, M. (2016). Towards devops in the embedded systems domain: Why is it so hard? In *2016 49th hawaii international conference on system sciences (hicss)* (pp. 5437–5446).

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*.

Macarthy, R. W. (2022, Sep). *Devopsimplementationmodel.* Retrieved from https://github.com/RuthMacarthy/DevOpsImplementationModel

Macarthy, R. W., & Bass, J. M. (2020). An empirical taxonomy of devops in practice. In *2020 46th euromicro conference on software engineering and advanced applications (seaa)* (pp. 221–228).

Macarthy, R. W., & Bass, J. M. (2021). The role of skillset in the determination of devops implementation strategy. In *2021 ieee/acm joint 15th international conference on software and system processes (icssp) and 16th acm/ieee international conference on global software engineering (icgse)* (pp. 50–60).

Machamer, P., & Silberstein, M. (2008). *The blackwell guide to the philosophy of science* (Vol. 19). John Wiley & Sons.

Mäkinen, S., Leppänen, M., Kilamo, T., Mattila, A.-L., Laukkanen, E., Pagels, M., & Männistö, T. (2016). Improving the delivery cycle: A multiple-case study of the toolchains in finnish software intensive enterprises. *Information and Software Technology*, *80*, 175–194.

Malone, T. W., & Crowston, K. (1994). The interdisciplinary study of coordination. *ACM Computing Surveys (CSUR)*, *26*(1), 87–119.

Mårtensson, T., Ståhl, D., & Bosch, J. (2018). Enable more frequent integration of software in industry projects. *Journal of Systems and Software*, *142*, 223–236.

Maxwell, J. A. (2012). *Qualitative research design: An interactive approach.* Sage publications.

McHugh, O., Conboy, K., & Lang, M. (2011). Agile practices: The impact on trust in software project teams. *IEEE software*, *29*(3), 71–76.

Merriam, S. B. (1988). *Case study research in education: A qualitative approach*. Jossey-Bass.

Meyer, B. (2018). Making sense of agile methods. *IEEE Software*, *35*(2), 91–94.

Mujtaba, S., Feldt, R., & Petersen, K. (2010). Waste and lead time reduction in a software product customization process with value stream maps. In *2010 21st australian software engineering conference* (pp. 139–148).

Myers, M. D. (1999). Investigating information systems with ethnographic research. *Communications of the Association for Information Systems*, *2*(1), 23.

Myers, M. D., & Avison, D. (2002). *Qualitative research in information systems: a reader*. Sage.

Nidumolu, S. R. (1996). A comparison of the structural contingency and risk-based perspectives on coordination in software-development projects. *Journal of Management Information Systems*, *13*(2), 77–113.

Noll, J., Beecham, S., & Richardson, I. (2011). Global software development and collaboration: barriers and solutions. *ACM inroads*, *1*(3), 66–78.

Nybom, K., Smeds, J., & Porres, I. (2016). On the impact of mixing responsibilities between devs and ops. In *International conference on agile software development* (pp. 131–143).

of Government Commerce, G. B. O. (2002). *Managing successful projects with prince2*. The Stationery Office.

Ohno, T. (1988). *Toyota production system: beyond large-scale production*. crc Press.

Olsen, C., & St George, D. (2004). Cross-sectional study design and data analysis. *College entrance examination board*, *26*(03), 2006.

Palvia, P., Midha, V., & Pinjani, P. (2006). Research models in information systems. *Communications of the Association for Information Systems*, *17*(1), 47.

Patton, M. Q. (2002). Qualitative research and evaluation methods. thousand oaks. *Cal.: Sage Publications*, *4*.

Perera, P., Bandara, M., & Perera, I. (2016). Evaluating the impact of devops practice in sri lankan software development organizations. In *2016 sixteenth international conference on advances in ict for emerging regions (icter)* (pp. 281–287).

Petersen, K., & Gencel, C. (2013). Worldviews, research methods, and their relationship to validity in empirical software engineering research. In *2013 joint conference of the 23rd international workshop on software measurement and the 8th international conference on software process and product measurement* (pp. 81–89).

Philips, Z., Claxton, K., & Palmer, S. (2008). The half-life of truth: what are appropriate time horizons for research decisions? *Medical Decision Making*, *28*(3), 287–299.

Pikkarainen, M., Haikara, J., Salo, O., Abrahamsson, P., & Still, J. (2008). The impact of agile practices on communication in software development. *Empirical Software Engineering*, *13*(3), 303–337.

Poppendieck, M., & Poppendieck, T. (2003). *Lean software development: an agile toolkit*. Addison-Wesley.

Proulx, A., Raymond, F., Roy, B., & Petrillo, F. (2018). Problems and solutions of continuous deployment: A systematic review. *arXiv preprint arXiv:1812.08939*.

Punter, T., & Lemmen, K. (1996). The mema-model: towards a new approach for method engineering. *Information and Software Technology*, *38*(4), 295–305.

Rembetsy, M., & McDonnell, P. (2012). Continuously deploying culture: Scaling culture at etsy. In *Velocity europe.*

Riungu-Kalliosaari, L., Mäkinen, S., Lwakatare, L. E., Tiihonen, J., & Männistö, T. (2016). Devops adoption benefits and challenges in practice: a case study. In *International conference on product-focused software process improvement* (pp. 590–597).

Robson, C., & McCartan, K. (2015). *Real world research, [argosy university].* West Sussex, UK: Wiley.

Robson, C., & McCartan, K. (2016). *Real world research: a resource for users of social research methods in applied settings*. Wiley.

Roche, J. (2013). Adopting devops practices in quality assurance. *Commun. ACM*, *56*(11).

Rodrigues, J., Ruivo, P., & Oliveira, T. (2020). Mediation role of business value and strategy in firm performance of organizations using software-as-a-service enterprise applications. *Information & Management*, 103289.

Rodríguez, P., Haghighatkhah, A., Lwakatare, L. E., Teppola, S., Suomalainen, T., Eskeli, J., ... Oivo, M. (2017). Continuous deployment of software intensive products and services: A systematic mapping study. *Journal of Systems and Software*, *123*, 263–291.

Rolland, C. (1998). A comprehensive view of process engineering. In *International conference on advanced information systems engineering* (pp. 1–24).

Rolland, C., & Prakash, N. (1996). A proposal for context-specific method engineering. In *Working conference on method engineering* (pp. 191–208).

Rowe, M., & Marshall, P. (2012). *The business case for collaborative devops.* Retrieved from https://www.ibm.com/developerworks/community/blogs/

Ro¨stlinger A, G. G. (1994). “generic flexibility—towards a component-based view of methods. In *Department of computer and information science, linko ¨ping university, linko ¨ping.*

Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering*, *14*(2), 131–164.

Saunders, M., Lewis, P., & Thornhill, A. (2019). Research methods for business students eight edition. *QualitativeMarket Research: An International Journal*.

Schumate, S. (2004). *Implications of virtualization* (Tech. Rep.). Technical Report 2004. www. dell. com/downloads/global/power/ps4q04-20040152 . . . .

Schuppenies, R., & Steinhauer, S. (2002). Software process engineering metamodel. *OMG group, November*.

Schwaber, K. (2004). *Agile project management with scrum*. Microsoft press.

Schwaber, K., & Beedle, M. (2002). *Agile software development with scrum* (Vol. 1). Prentice Hall Upper Saddle River.

Sekaran, U., & Bougie, R. (2016). *Research methods for business: A skill building approach*. john wiley & sons.

Senapathi, M., Buchan, J., & Osman, H. (2018). Devops capabilities, practices, and challenges: Insights from a case study. In *Proceedings of the 22nd international conference on evaluation and assessment in software engineering 2018* (pp. 57–67).

Shahin, M., Babar, M. A., & Zhu, L. (2017). Continuous integration, delivery and deployment: a systematic review on approaches, tools, challenges and practices. *IEEE Access*, *5*, 3909–3943.

Sharp, H., & Robinson, H. (2008). Collaboration and co-ordination in mature extreme programming teams. *International Journal of Human-Computer Studies*, *66*(7), 506–518.

Siqueira, R., Camarinha, D., Wen, M., Meirelles, P., & Kon, F. (2018). Continuous delivery: Building trust in a large-scale, complex government organization. *IEEE Software*, *35*(2), 38–43.

Slama, D., Puhlmann, F., Morrish, J., & Bhatnagar, R. M. (2015). *Enterprise iot: Strategies and best practices for connected products and services.* " O'Reilly Media, Inc.".

Smeds, J., Nybom, K., & Porres, I. (2015). Devops: a definition and perceived adoption impediments. In *International conference on agile software development* (pp. 166–177).

Society, T. B. C. (2012). *Sfia view - devops skills.* Retrieved from https://sfia-online.org/en/tools-and-resources/sfia-views/devops-skills-in-sfia

Ståhl, D., & Bosch, J. (2013). Experienced benefits of continuous integration in industry software product development: A case study. In *The 12th iasted international conference on software engineering,(innsbruck, austria, 2013)* (pp. 736–743).

Ståhl, D., & Bosch, J. (2014). Modeling continuous integration practice differences in industry software development. *Journal of Systems and Software*, *87*, 48–59.

Ståhl, D., Hallén, K., & Bosch, J. (2017). Achieving traceability in large scale continuous integration and delivery deployment, usage and validation of the eiffel framework. *Empirical Software Engineering*, *22*(3), 967–995.

Stapleton, J. (2003). *Dsdm: Business focused development.* Pearson Education.

Stol, K.-J., Ralph, P., & Fitzgerald, B. (2016). Grounded theory in software engineering research: a critical review and guidelines. In *2016 ieee/acm 38th international conference on software engineering (icse)* (pp. 120–131).

Sverrisdottir, H. S., Ingason, H. T., & Jonasson, H. I. (2014). The role of the product owner in scrum-comparison between theory and practices. *Procedia-Social and Behavioral Sciences*, *119*, 257–267.

Sydow, J., Lindkvist, L., & DeFillippi, R. (2004). *Project-based organizations, embeddedness and repositories of knowledge* (Vol. 25) (No. 9). SAGE publications Sage CA: Thousand Oaks, CA.

Talwar, V., Milojicic, D., Wu, Q., Pu, C., Yan, W., & Jung, G. (2005). Approaches for service deployment. *IEEE Internet Computing*, *9*(2), 70–80.

Ter Hofstede, A. H., & Verhoef, T. (1997). On the feasibility of situational method engineering. *Information Systems*, *22*(6-7), 401–422.

Van de Ven, A. H., Delbecq, A. L., & Koenig Jr, R. (1976). Determinants of coordination modes within organizations. *American sociological review*, 322–338.

van Slooten, K., & Brinkkemper, S. (1993). A method engineering approach to information systems development. In *Information system development process* (pp. 167–186). Elsevier.

Vlietland, J., & van Vliet, H. (2015). Towards a governance framework for chains of scrum teams. *Information and Software Technology*, *57*, 52–65.

Wahaballa, A., Wahballa, O., Abdellatief, M., Xiong, H., & Qin, Z. (2015). Toward unified devops model. In *2015 6th ieee international conference on software engineering and service science (icsess)* (pp. 211–214).

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., & Schulz, T. (2017). Key capabilities of devops teams and their influence on software process innovation: a resource-based view. In *23rd americas conference on information systems.*

Williams, C. (2007). Research methods. *Journal of Business & Economics Research (JBER)*, *5*(3).

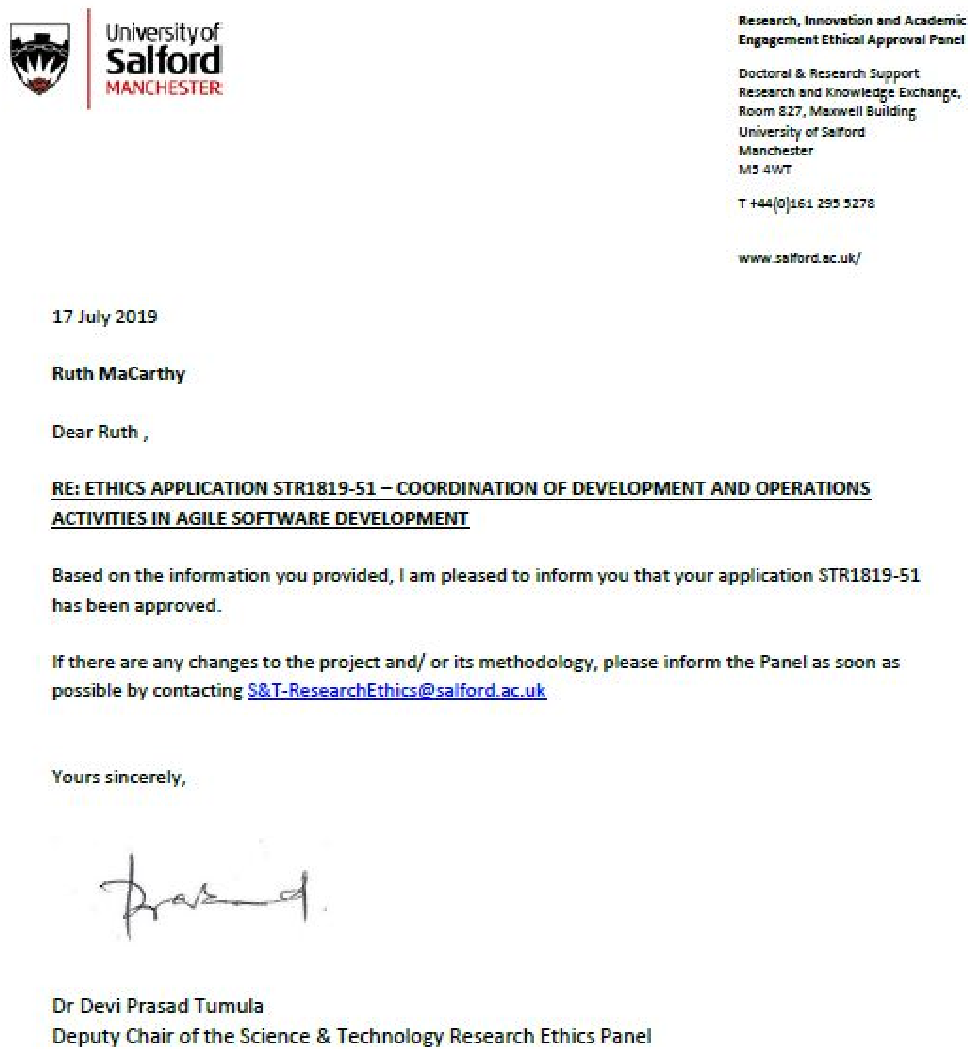
Willis, J. (2010). *What devops means to me.* Retrieved from https://blog.chef.io/what-devops-means-to-me

Yadav, R., Mittal, M., & Jain, R. (2018). Adoption of lean principles in software development projects. *International Journal of Lean Six Sigma*.

Yu, X., & Petter, S. (2014). Understanding agile software development practices using shared mental models theory. *Information and software technology*, *56*(8), 911–921.

Appendix A

Ethics Approval


University of Salford MANCHESTER

Research, Innovation and Academic Engagement Ethical Approval Panel

Doctoral & Research Support
Research and Knowledge Exchange,
Room 827, Maxwell Building
University of Salford
Manchester
M5 4WT

T +44(0)161 295 5278

www.salford.ac.uk/

17 July 2019

Ruth MaCarthy

Dear Ruth ,

RE: ETHICS APPLICATION STR1819-51 – COORDINATION OF DEVELOPMENT AND OPERATIONS ACTIVITIES IN AGILE SOFTWARE DEVELOPMENT

Based on the information you provided, I am pleased to inform you that your application STR1819-51 has been approved.

If there are any changes to the project and/ or its methodology, please inform the Panel as soon as possible by contacting S&T-ResearchEthics@salford.ac.uk

Yours sincerely,

[illegible]

Dr Devi Prasad Tumula
Deputy Chair of the Science & Technology Research Ethics Panel

Appendix B

Interview Guide

Coordination of Development and Operations Activities in Agile Software Development: Semi-Structured Interview Guide

1. **Background Notes** I want to ask you about your experience of agile practices and the coordination of work between development and operations in software development process. The research involves interviews with people doing a range of different roles and from different companies which may have contrasting strategies.

The purpose here is to try to understand the interactions and dependencies between development and operations activities in organizations, so that we can try to learn for the future. This is being done as part of a PhD research in the University of Salford. I want to ask you the following questions and tape record your answers. Can I switch on the recorder?

2. **Organisational structure and role Identification**

Could you please describe your organisation and its structure?

Please describe your current role?

What are the other roles involved in your software development process?

Please describe the position of your team in relation to other teams in your organisation.

3. **Development and Operations Coordination**

Please describe a software development project you were involved in recently?

What was the scope of the projects (budget? Number of people? Duration?)

What was your role in the project?

Can you please describe the work flow from development to delivery of the project?

How were tasks assigned on the project?

Please describe the software development tools used for the coordination of work on the project?

Probing Questions

Build tools

- Automated testing tools
- Automated deployment tools

Development Frameworks

Please describe the manual activities involved in the project.

Could you please describe typical causes of delay in your projects?

How were incidents and unplanned work handled during the project?

How was commit organised? (How was work coordinated on the same part of codebase?)

What challenges did you face with the coordination of work on the project?

What do you think could have improved the coordination of work on the project?

What organisational factors (such as staff skills and budget availability) affected the coordination of work on the project?

What infrastructure issues (such as hardware, operating systems and development/test platform availability) affected coordination of work on the project?

If you could fix one thing about your current software development process, what would it be?

4. **Software Development Process**

Please describe the development process your organisation practice? Probing Questions

- Incremental development
- Sprints/iterations
- Daily Stand-ups
- Product demonstrations
- Retrospectives
- Configuration management
- CI/CD
- Continuous Deployment

5. **Code Deployment**

Please describe your approach to code deployment.

Please describe the software technologies you are using.

Probing Questions

- RESTful APIs (web services)
- Deployment environment (Cloud /On-prem)
- Programming languages
- Database Technologies
- Legacy systems

6. **Open-Ended Question**

Is there anything else you think is relevant we have missed?

Is there anything else you think we should have asked?

7. **Personal Details**

What is your name?

What is your job title?

How long have you been working at your current organisation?

How long have you been working in the industry?

Do you know anyone else I could talk to about this study?

Can I talk to you again in a month's time ?

8. **Note** The interview guide will need to be tailored to specific respondents. We do not need to ask someone who is not a developer of the build tools used in their projects. Instead we might want to ask about their involvement in automated deployment. Our approach is to develop a somewhat bespoke interview guide, depending upon the role of the interviewee.

Appendix C

Consent Form

Computer Science and Software Engineering
Newton Building
University of Salford
Manchester
M5 4WT
UK
Email: r.w.macarthy@edusalford.ac.uk

<u>Participant information and consent form</u>

Thank you for agreeing to take part in an investigation into the patterns of work coordination between development and operations in software development projects and collaboration. This is a PhD research being carried out in the University of Salford. This Participant Information Statement and Consent Form explains what will happen if you choose to take part in this evaluation, so you can make an informed choice about whether or not to take part. Participation is voluntary.

What will I be asked to do if I participate? Meetings will normally be conducted by telephone or Skype for up to 45 minutes and by direct observation of activities if giving the opportunity.

What will happen to the information I provide? The meeting will be recorded with your consent (audio only) and transcribed. You have the option (below) to remain anonymous or allow your name to be associated with the data you provide. Data will be stored securely for 7 years before being destroyed, and will only be used by the team conducting the investigation at Computer Science and Software Engineering at University of Salford. You can request a copy of the data. Data will be analysed and may be used in a qualitative empirical study for a PhD research. If you would like to receive a copy of the qualitative study when it is published, please indicate this below (however, note that the research will be made public).

What if I want to withdraw from the study? If you do consent to participate, you may withdraw at any time by contacting Ruth Macarthy. You are free to leave the interview discussion at any time. You may also refuse to answer any questions that you do not wish to answer during the interview.

Who is conducting the study? The study is being led by Ruth Macarthy under the Supervision of Dr. Bass of Computer Science and Software Engineering at University of Salford.

Please indicate your consent and sign overleaf.

Please tick all, if you agree:

- ☐ I have read the Participant Information Sheet and understand the purpose of the research
- ☐ I am over 18 years of age
- ☐ I freely agree to participate in this study as described and understand that I am free to withdraw at any time during my interaction with the study.

Please tick one:

- ☐ I consent to being referred to by name in the qualitative empirical study and any other publications relating to the study; or
- ☐ I consent to being referred to by my place of work and title in the qualitative empirical study and any other publications relating to the study; or
- ☐ I consent to the information I provide being used for the purposes of the aforementioned study only if it is fully de-identified (anonymized)

Optional:

- ☐ I would like to receive a copy of the qualitative empirical study when it becomes publicly available

Name of Participant (please print):____________________________________

Signature of Participant:____________________________________

Date:____________________________________

Declaration by Researcher:
I have given a verbal explanation of the study, its activities and risks and I believe that the participant has understood that explanation.

Researcher Signature:

Date: ____________________________________

Appendix D

Participant Information Sheet

Coordination Of Development And Operations Activities In Agile Software Development

Participant Information Sheet

August 2019

Ruth Macarthy
Computer Science and Software Engineering

1. Research Project Title

Coordination Of Development And Operations Activities In Agile Software Development

2. Invitation

You are being invited to take part in this research project. Before you decide to do so, it is important that you understand why the research is being conducted and what it will involve. Please take the time to read the following information carefully and discuss it with others if you wish. Ask us if there is anything not clear or if you would like more information.

3. What is the Project's Purpose?

The purpose here is to try to understand the interactions and dependencies between development and operations activities in organizations. We will study industry best practices and make informed recommendations to improve work coordination patterns in the software development process for the future. This is being done as part of a PhD research in the University of Salford.

4. Why Have I been chosen?

You have been chosen because you are a practitioner, or aspiring practitioner, with specialist skills and knowledge in software development and delivery activities.

5. Do I have to take part?

It is up to you to decide whether or not to take part. If you do decide to take part you will be able to keep a copy of this information sheet and you should indicate your agreement on the consent form.

6. What will happen to me if I take part?

You will be asked to participate in an interview about your experiences in software development processes which we estimate will take 45 minutes.

7. What do I have to do?

Please answer the questions in the interview. You will be asked to give responses based on your experience or your impressions.

8. What are the possible disadvantages of taking part?

Participating in the research is not anticipated to cause you any disadvantages or discomfort. The potential physical and/or psychological harm or distress will be the same as any experienced in everyday life.

9. What are the possible benefits of taking part?

Whilst there are no immediate benefits for research participants, it is hoped that this work will have a beneficial impact on how work is coordinated between development and operations in a software development project. Results will be shared with those participants who request it on the consent form, in order to inform your professional work.

10. Will my taking part in this project be kept confidential?

You will have the choice for your responses to be kept confidential, depending on the permission you grant on the consent form.

I. You can be referred to by name in the qualitative empirical study and any other publications relating to the study if you consent to; or
II. You can be referred to by your place of work and title in the qualitative empirical study and any other publications relating to the study if you consent to; or
III. The information you provide for the purposes of the aforementioned study will be fully de-identified (anonymized), if you wish.

11. Will I be recorded, and how will the recorded media be used?

Interview participants will be audio recorded during the interview. The audio recording will be transcribed verbatim into a script describing your words. Audio recordings and transcripts will be kept on password protected University of Salford servers These words will be analysed and maybe quoted in reports and publications. The quotes will be kept anonymous or attributed to you or your affiliation depending on the permission you grant on the consent form.

12. What type of information will be sought from me and why is the collection of this information relevant for achieving the research project's objectives?

The questions will be about your opinions and current practices in relation to software development activities. Your views and experience are just what the project is interested in exploring.

13. What will happen to the results of the research project?

Results of the research will be published. You will not be identified in any report or publication, unless you have given us permission to do so on the consent form. Your institution will not be identified in any report or publication, unless you have given us permission to do so on the consent form. If you wish to be given a copy of any reports resulting from the research, please ask us to put you on our circulation list.

14. Who is organising and funding the research?

This research is being conducted by Ruth W. Macarthy as a PhD study in the University of Salford, with funding from Petroleum Technology Development Fund (PTDF).

15. Who has ethically reviewed the research?

The research has been reviewed and approved by the University of Salford Ethics committee with reference number **STR1819-51**.

16. Contacts for further information

Ruth Wakeni Macarthy
Computer Science and Software Engineering
Room 150, University of Salford
Newton Building
The Crescent
Manchester, M5 4WT

Mobile: +44 (0) 7521621425
email: r.w.macarthy@edu.salford.ac.uk

Dr Julian M. Bass CEng FBCS CITP SFHEA
Computer Science and Software Engineering
Room 218, University of Salford
Newton Building
The Crescent
Manchester, M5 4WT

Telephone (external) +44 (0) 161 295 2883
Telephone (internal) ext. 52883
email: j.bass@salford.ac.uk

Prof Sheila Pankhurst
Dean
School of Environment and Life Sciences
G41, University of Salford
Peel Building
The Crescent
Manchester, M5 4WT

Telephone (external) +44 (0)161 29 55171
Telephone (internal) ext. 55171
email: s.pankhurst@salford.ac.uk

Appendix E

Interview Transcript Sample

File: INT_3.MP3

Duration: 41:41

Date: 29/09/2019

Typist: Interviewer

Interviewer: So, I have the recorder on.

DvOps1: ok

Interviewer: Thank you for agreeing to participate in the study.

DvOps1: No problem

Interviewer: I would like to understand what your organisation does and how it is structured. Could you describe your organisational structure and what your organisation does?

DvOps1: OK, so, I currently work at ING. Probably familiar. It's a big bank in

Netherland but they also have different departments and also departments through out the world. Em... they work, like the spotify model as we say it. So it's potentially tribes and all these emm... teams. So it's... I'm part of the cloud team, and this basically is a new team which was launched a couple of months ago. And the goal of the team is to investigate the movement to the public cloud. As you might be aware.. em... banks have regulators and regulators can be nasty...And so whenever you host client data or whatever kind of data which is confidential... em.... Regulators would tend to see that you .. come to the encryption and all these parts. And the public cloud is perceived as not safe in this case. So we're trying to see which movement and tools we need to use in order to be able to use the public cloud. Em... and what we've actually done is eh.. say ok, why don't we don't we just say the public cloud is safe and and just deploy our services and see what the regulators have to say. So we're gonna turn it around and we're just gonna create this whole infrastructure environment into the public cloud and then see which applications with different security levels we can host on this public cloud environment. And this is will actually allow us to do faster development, em.. you know create new apps. Be able to launch them faster because, eventually, we do everything in codes, that's our goal. And without creating the infrastructure on-premises. So this is the goal of the team at this point. Is that clear?

Interviewer: Yes, it is. Thank you.

DvOps1: Ok

Interviewer: And um, can you describe how your team relates with other teams? Like what is the relationship between your team and all the other teams in your organisation?

DvOps1: Ok, this is also interesting because ING has a lot of teams .. a couple of hundreds and so it's achieved as em... I cannot say it's hard, but we need to make ING aware of the fact that we are creating this public cloud. And we kind of have to just lobby people that we have. They go out to different teams and different managers and tell them, hey listen. You know we have this cool thing. We work with Kubernetes. We would like your team to be part of it. So it's kind of con...

we have these consultants within our team, that are part of our team, and they reach out to other team managers and ask them if they are willing to participate into this new public cloud . . . xx that we're actually creating and trying to on-board them. So what we do is to give them the right tools, we have this development environment and tell them, ok, let's see if you can use this pipeline, DevOps pipeline as we call it in Azure and see if we can on-board your team into a containerized platform. So basically, I think it's kind of a lobby that we're doing at the moment.

Interviewer: Ok, and these other teams, are they development teams?

DvOps1: Yes, they're purely development teams. They create APIs, beacon APIs or a full term applications or just parts of websites. It's kind of different teams who do software development.

Interviewer: Oh, ok. And what is your role on this team?

DvOps1: So, em.. my role on this team is, I am em. . . as a consultant responsible for the automation and creation of the infrastructure part. So if you look at the lowest level where you have network and security, as x.. tools like application gateway, firewalls but also Kubernetes, as a cluster. I'm sure you're aware of Kubernetes and Docker containers?

Interviewer: Yes

DvOps1: Em.. so, this whole infrastructure stack we create by using terraform xx in which we can use and deploy infrastructure as code. So, I create all the terraform scripting and modules and I glue everything together. And we also build pre-fab pipelines for our customers so, the pipeline yamls and everything. So, all we want to do, all we want the teams to do actually, write their codes and put it in a repository, hook up our pipeline yaml and then the application should be container-aware of course, and then it should flow into our platform. So, this is everything I've built together with 4 other people.

Interviewer: Wow, interesting. Em. . . how many people are on that team?

DvOps1: If you're talking about my team? I think we're 5. 5 People. I think 5 technical people and the of course your managers and other guys more into management. But in technical, I say em.. 5 techies... 5.

Interviewer: OK, so what are the roles of these other persons in the group?

DvOps1: Em...it's combined. So basically because I'm external, so eventually, I will go. So we do a lot of knowledge transfer but all 5 do the same. Of course we have different tasks throughout the sprint that we have. So I'm more specialised in terraform with other guys who work more on the pipeline stuff or we also use helm for instance for helm charge. Em, so we divide the work a little bit but it should be rotating. Everybody should understand and know what the other person does eventually. So it's.. so yeah we can exchange.

Interviewer: Ok, so are all these team members DevOps engineers or they are developers?

DvOps1: No we don't actually have developers in our team. So in our case... it's just DevOps. I... I'm not sure if I like the word DevOps but of course, it's the trend as we say now but us five people are responsible for creating this cloud, public cloud solution. That's how I wanna say it. And we're trying to on-board the software development teams, all these different teams who are actually writing the codes, to be able to support them and host their applications on our platform. So we're the platform builders. X.. the cloud builders. Maybe that's a better eh... because of ... eventually I do develop for myself as well, so I know how the development process works and how you then .. (breaks.. phone rings)... so eventually, I understand the development process but in this case, we are just on the infrastructure part and a little bit on the development part, it's the pipeline part that we create... yes.

Interviewer: Interesting. With regards to your interaction with Developers, have you concluded any projects?

DvOps1: Eh... we're actually busy with two development team, we're onboarding. . .

Interviewer: Ok

DvOps1: Eh. . . but what we try to do is, we try to give them the right tools. So that basically means that we define the pipeline . . . we were talking about pipelines. And the only thing that we actually want them to do is write the code and we should know how to start your application. And they create the docker file which comes with it. And we try.. from there we try to pick up the rest. So we try to create the image automatically through DevOps xxx or the pipeline, we then host it in our container registry. We then create the specific yamls to be able to install and run this applications on Kubernetes and we also try to create the .. roles or the virtual services used ..xx..xx..xx to be able to route the application. And we do this by using helm. Eh. . . Helm is eh. . . helm charting. So we try to let them fill the correct values, because helm uses values . . . files. . . Em, by explaining to the how it works and trying to fit their application into this helm chart, we were then able to actually deploy any application that we want. So this is the way that we try to achieve it, yes.

Interviewer: Ok. If you don't mind, I think I'll understand it better if you explain this using a project. So if you just take a project and explain from it's inception to when it's live on the cloud environment.

DvOps1: So we recently took the list ad, it's kind of a static page or a static website with some information. And so we joined with the team and we told them how we're actually working. And together with them, we tried to, we need to adjust the application because it was not container-aware. So, together with them, we altered the code a little bit so it was container-ready. And the we also tried to fill the application and the value file for them. And then basically, when that was done, we created a pipeline in DevOps and then we just deployed the application so in... yeah, those were the steps that we took in this case.

Interviewer: Ok, so you deployed the application, or the development team deployed it?

DvOps1: Eventually, in the first time, we did it ourselves, but when the development pipeline is ready, they don't need us anymore. Because then, they'll be able to change the value cells themselves and based on the values, they would get different output but the whole development pipeline will be fixed and static. And they don't need to change it anymore, they don't need us anymore.

Interviewer: Ok

DvOps1: Yeah Interviewer: So, when you're done, your business with them is ended?

DvOps1: Yes. In... in... in some way, yes. Of course, because it's all new technology, a lot of developers are not aware or not familiar with the way it works. So what I see happening is when you're trying to automate all these processes for them, or actually say to them.. hey, listen. You don't need to be aware of where your application is running. All you have to do is fill this file and make sure your application is docker-aware. We would do the rest. And it seems that in some cases, they find this a little tricky because they're actually out of control where a lot of developers... they tend.. you know in the past they would ask for a virtual machine with 10Gb of memory, I want 3 CPUs and I need 200Gb of disk space and ... to rollxxxx. So what we're trying to do with this, we're trying to take this part away and say, listen, you're a software developer, you write codes, we do the rest. And of course there's the gray area where you need to teach them how the transition form codes to infrastructure works. I can see that whenever it's clear, they would eventually see that ok, we don't have to mind, you know. It's done automatically for us. Which is pretty convenient because, we like to write code. That's what they do. And I think they should not be aware of the fact of how it work or where it runs or how many memory it needs. Of course it important that they write their code as optimal as possible, but I don't think they should be... they should be asking for virtual machines or resources. I think that's part of the DevOps team in this case, because we should understand developers a little bit and everything that comes under it in operations. So, eh.. yes... it's emm... a learning curve they have to go through and they need to let go. Because they still ask for, can I get access to this cluster, can I… basically, that's what we want to get rid of. They shouldn't have access anymore because they don't need it.

Interviewer: So if the developers are done with one project and need to proceed to another, what happens?

DvOps1: Umm… eh… of course it is out of our line-sight so, this basically means it happens outside of our…our…spectrum, but I mean, they can just start a new project and basically, we have them um… demo repository, which they can pull, and then just inject their code, write the value for it and directly deploy. Because what we do is…depending on the team they are part of, they have access to a certain name-space in our Kubernetes cluster. So if they start a new project, they use the demo repo, they write their code and the basically fill the yaml file, the values file. And then they just use the same pipeline, only.. of course, with a different repository and the application will be deployed in the same cluster, in their own namespace.

Interviewer: Oh, ok. So who manages the application when the code is deployed.

DvOps1: Ok, good question. So what we have is.. we haven't … we try to facilitate them with an …xxx… so this is basically a second Kubernetes cluster, which er… on all the working clusters, we have fluent-bit daemons running as a daemon set. And we have Prometheus scrapers. So what we do, we try to collect all metrics and all .xx.. where we have..x.. Kibana and …xx.. where we have Prometheus extra, we have Kiali for meshing tracking and we're trying, we're still developing this, so bear with me, because we're also still in this research phase, but we try to facilitate the with low output so they read all these loggings that the application produces, it might be errors or whatever, but also the metrics that they expose, they just expose the metrics endpoints and we have Prometheus scrapers which scrapes all their metrics and they should be able to create their own application dashboard. And so this is basically how we then try to visualize and monitor their application.

Interviewer: OK, would you know the processes for the deployment of the applications. Because you're basically involved in the infrastructure part.

DvOps1: So, in azure, we have what is called DevOps, and Devops allows you to create pipelines. And I can just see if I can open one of the pipelines here locally. So basically, we pull in their code, because it's all git or in the case, repo-based.

Interviewer: Ok

DvOps1: I'm just gonna open the repo here, let me see, my demo repo. So we have this pipeline, So I'm just gonna open the pipeline as we have it now. So, we have our own azure container registry, as in a repository outside the cluster. So what we basically do is, we fetch the log in, … in the azure container registry. Then we have some xx… scripting which is just a docker building. The docker tagging and a docker pushing. Based on that… er…we run the helm installer, em… like I said we use helm charts. Which is a description of the application, how it's run, what variables and how it's started so we actually inject that within helm install or actually in this case, we template them first. And then, what we do afterward is we copy the artefacts so we can use them in a later state as well. We publish the artefacts, and eh… because these templates are generated as yaml files, we can then connect to Kubernetes to the cube API and deploy the yaml files so there's a deployment. Kubernetes deployment, Kubernetes services, they're all part of the output of the template and they're just deployed to our cluster. So these are actually the steps we take.

Interviewer: Um..hm…

DvOps1: Is that clear?

Interviewer: Yes, it's clear but I'm just wondering what the developer has to do when they'll need to release a new feature or something..

DvOps1: Yeah… so basically, it's hooked up to a commit. So, the DevOps pipeline is as a hook. So whenever you commit to master in your git repository, this hook is automated and initialised so basically, the pipeline starts running. So it will pull .. in the latest code, compile it, test it, create docker image. Push it to the ACR, helm template, output yaml file and deploy the new deployment referring to the latest image of the docker repository. That's how it works.

Interviewer: Oh, ok

DvOps1: Yeah

Interviewer: And eh… are there any manual processes involved in this project?

DvOps1: No, eh… in this particular process, no. There is no manual intervention. There is only one manual intervention at this moment and it's the creation of the service connection as they call it. So it's a bit technical story, but umm… whenever a new team on-boards, they would need their service account credentials to be hooked into this service connection. So the Azure DevOps pipeline is able to connect to our Kubernetes cluster. So this is a one-time manual process at the moment but we're trying to automate this as well.

Interviewer: Ok

DvOps1: Yeah

Interviewer: And do you have manual testing done?

DvOps1: Emm… no. The truth… not be the case eh… we encourage.. eh actually… I think all developers should write their tests in an automated way. So this can be part of the Devops pipeline. So whenever you build a software, if it compiled or not, depending on what kind of code they write, they also should compile their test with it and also should run their tests. And if the test fails, the pipeline fails then they're owner of the pipeline so they can see what fails.

Interviewer: Ok, so basically, you do not have anyone monitoring the quality of the software being developed by specific groups of developers?

DvOps1: Oh we have different tooling for that. So we have coding tools which analyse codes for repetition… I know… I'm not really into that part but I know there's automated code testing

Interviewer: Ok

DvOps1: And ..but..we can make that part of this pipeline as well, so if there's an endpoint or whatever, we can just ask you if this code is any tech debt or whatever, what's the code's standard or the coding em.. emm.. I cannot find the word for it…but there's eh.. the code coverage. So, there are these tools, we have eh..what's it called? Phonics… you have a name… which you can read or test your code coverage and stuff. So there's external tooling which you can use to eh… and integrate into your pipeline. We don't have that at the moment. But we sure could do that. There's no problem.

Interviewer: UM… Ok. Could you describe any typical delays you have in your projects?

DvOps1: Delays that we have… ok yes… so yes we have some tech debt when it comes to terraform. There's been a major upgrade…so we need to upgrade all our terraform scripting to meet the new version. 0.12 actually. It's been out for a little while now but um… umm.. we're finally making the transition to the new version, um…also we did a lot of testing on a different Kubernetes version. We still have a service mesh which we need to test in the proper way. We're now currently using Istio but there's a lot of other solutions as well like Linkerd but also .. is creating a new service mesh as well, which is very good. So we're still debating on what the best solution is for our service mesh. So in this case, we do have some delays on that part. Because it's you know this movement is just going so fast, these tools exploding but sometimes you just have to make a choice and say, ok we're gonna use this product the upcoming year and we can see afterwards, yes.

Interviewer: Ok. So what would you say is the major difference between your team and all the other software development teams in your organisation.

DvOps1: Emm…Well, I mean the difference between the software development team and er. . . If you’re referring to the teams within Ing, I guess?

Interviewer: Yeah

DvOps1: It’s that we have the luxury of being greenfield so, we could actually start from scratch, which is also very nice. And we also have er. . . have I guess the luck that we’re not bound to the ING network, because we’re working in the cloud. So that basically gives us a lot of freedom, so we’re not tied to rules as much as some other software development teams are at the moment. So I guess that’s a big difference for us. Which allows us to you know, think more out of the box. And we have more freedom. ING also has a private cloud for instance. You can see that’s a lot more harder, the have a lot more rules. . . and certain..xx. . . to be aware of. So I guess, that might be the biggest difference at the moment, that we’re running as public cloud.

Interviewer: So how do you integrate with legacy systems?

DvOps1: So. . . there’s a migration path and basically, this comes down to the development team. Emm. . . so if the development team is willing to host their application, which was a legacy application and wants to host it on the cloud, they would have to do the work. So, we are not responsible for that. So we write the tools and help them to migrate. But if there’s legacy emm.. for instance xxxx which is bound to libraries that we cannot offer, eh. . . then they’re responsible and is also probably the pressure of the bank that would come in play as well. So if this application really needs to move forward, then they will have to free up the developers to, you know, to re-write codes or make sure the code is container-aware. So we are not integrated into that part of the process.

Interviewer: I know there are Devops teams with mixed responsibilities, and some others that try to bridge the gap between development and operations. But I feel your responsibilities is more like automated Ops. What do you think about that?

DvOps1: Yes, I guess, in some ways, it is. I think that a.. a. . . good view of what we’re doing actually. So of course, we develop in our operation environment, so we do develop the whole infrastructure part with the AKS cluster automated but eventually, yes it's all automated so, we tend to.. because we either own ..xx. . . xxx.. so, this pipeline I described to you, which we take the code and we deploy it on our system, we use the same pipeline, only different steps to deploy our own infrastructure. So our whole infrastructure code is injected into our pipeline as well. So we can just run the pipeline, and say create new infrastructure and it would create the whole infrastructure with the application . . . with the AKS cluster or Kubernetes cluster and everything that comes with it, with all the observabilities of. . . with one push of a button in this case, the Azure development of DevOps pipeline.

Interviewer: Yeah. So as you provision an automated infrastructure, for a particular group, you really have nothing else to do with the group?

DvOps1: No..no. . . not in a way. Not in that way. Of course we try. . . in the beginning, we try to onboard them and that basically means we also need to look into their code, to help them but our responsibility does stop there. We not responsible for providing nice-written and optimised codes.

Interviewer: If they're to back up their codes, how does that work?

DvOps1: As in backups? What do you mean by that?

Interviewer: Like general backups, like the backups done for data centres.

DvOps1: Oh, sorry, back ups. Yeah. Um. . . so what we try and achieve is first, we try to achieve the migration of stateless applications. This is or primary goal. So we are trying to onboard teams which has as less state as possible. And if there is state in some way, we do have a challenge. But what we try to do is if ti's database related, we try to use the azure provided databases, relational databases for instance or whatever suits them best and we try to do as many manage service as possible. Now this is where the trick comes, because if you take the regulators and you have hosted data in the cloud. . . which kind of data is this? Is this no. is this safe?... then you would just save this data? Of course the managed service also provides backup facilities, which we can then again automate with terraform. This is definitely a must for us… that it should be managed in an automated way. Em.. we do still have the option of provisioning physical volumes. Kubernetes has the option of doing physical volume claims to a cloud provider. And in this way, we can based on the application, we could provision 2 disks and one backup disk in one location and the other one in the other so, we can, you know synchronise data in some way. But…I must say it's still then a good communication between our team and er.. the development team. So they need to make clear what their wish are in this case, or if they wish to store data, what kind of data is it? How much is it and eh…and we can provide them with a suitable solution, either through a shared storage or a physical volume claim or something else. But for this moment, now that we're in the transition of actually onboarding teams, it's stateless first.

Interviewer: Ok, interesting. What would you say are the challenges you faced during this process?

DvOps1: Emm.. well, I think when you trying to build something from scratch, you don't have all the boundaries set yet, so it's what happens is, also with all these new tooling coming and…new versions, and em…maybe we run into some .xx.. for instance, you… especially when stuff is new, you need to find out…xx.. because not everything is documented, and you cannot read everything. So I guess, the best thing, the challenge is to set up an environment which is suitable for the whole company and this is actually pretty hard because there's different teams with different wishes, and you soon realise that the environment you created is not suitable enough or its too small or its, we're missing this or we're missing internal DNS or we should em… or virtual xxx service should work in a different. So, it's kind of ummm…trial and error in some ways. Whereas the basis is fine but once you try to extend you need to be aware of always looking to the core again. Does it not break the core or whatever? So we need to redeploy cluster sometimes a couple of times because we need to make additional changes so low in the code that it basically means we need to redeploy the infrastructure. And that basically means you also need to redeploy the application you are running on it. So yeah, We had this challenge a couple of times. But I think, by recreating everything through pipelines, and making your code redeployable, you keep on testing this process over and over again. So eventually, you should not be afraid of destroying stuff because you know for a fact that you can recreate it.

Interviewer: Have you had any challenges with the other teams?

DvOps1: Em... yes. So this especially has to do with the fact ... like I explained that sometimes, they're used to having a lot of access and we need to try to tell them that you know, in this environment, it's just not gonna happen. So they should be aware of the fact that we're taking away a lot of their access that they used to have. But in return, providing them with tools, which allows them to easily deploy on infrastructure. So, I think that's also a problem that we need to raise them again, or teach them, or make them aware....there is benefit for the bank, so not for the person but for the whole company and this is also a mindset that whenever you're working in a company, you're part of the team. Of course you're working there for yourself or for your own development. But you should keep an eye on the company's perspectives not because you're doing this for the company, you know, and by doing this kind of tooling and using it and give them the tools that they need, it's actually easier for them to deploy. So sometimes we, it's hard to explain and make them aware of the fact that we're not doing this to tease them, but we're doing this to make life easy for them. So I think that's also a big challenge that we're facing at the monent with the teams. And also we have teams who are just not ready to be containerised, so they're still in eh, mmore monolithic application and em.. many you'ld like to... legacy, and they're just now making the step towards containerization and you actually need to help these people also... how the process works..explain to them that we have all kinds of security rules.. this... ports of security.. how this pipeline works. I also feel that ther's a knowledge gap as well, sometimes.

Interviewer: Ok, em... in the event that you're able to onboard all the teams in ING, what would be the responsibility of your team after that?

DvOps1: Emm. I think my responsibility is to lock the door and go away. Eventually, emm... I think whenever this infrastructure is suitable for production, and we can run ING applications and there's connectivity and all these stuff, I guess my work will be done.. emm… I think there's still … of looking of see if there will be a management team, who'll actually gonna manage these clusters, and the … that we're actually written so far. I'm not sure we will be responsible in the future for the actual production environment so we…I'm not sure… they're still debating in ING as well what the best approach would be. But..I don't think there's an easy answer to that. In some way, it would make sense for us to be responsible, but in other ways, it's also… we have teams who are standby and all these stuff.. so, um.. I think they're still debating on that part.

Interviewer: And for the on-prem systems, do you have system admins?

DvOps1: Yes, yes we have different infrastructure teams, actually. So we're the first public cloud team but there's a lot of infrastructure teams in ING as well, who work on the… the private cloud and they have legacy systems within the ING. So yes…w there's a lot of overlap actually. So there's a lot of skill sets available in Finco1 but within different teams. Em.. so there's actually… that would be an interesting development because eventually, if the public cloud is allowed, there would be a movement from internal to the cloud and I'm pretty sure that em.. some people, some roles will disappear but teams would grow, especially the teams that we're working at the moment so, I do see a shift at some point. And this is has to do with the fact that we need connectivity to the inside ING network. It's a very long process especially with security. But once we have that solidated and we are allowed to do a lot of stuff and we're fine, then yeah, we'll probably see a move. Yes. But eh… I'm… I might not be part of that because eventually I'm still into consulting so I work externally but do see a bigger demand or move into the public cloud because of the tooling.. because of the speed of the development. Probably within one or two years, there would a lot different within ING.

Interviewer: If there's something you can change, what would that be?

DvOps1: Well, that's a good question. Well, em…the thing is I don't think at this moment there is anything to change because we have a greenfield situation and that basically means that we set up the team and the environment as we want it. In our case, our team is pretty new. It's been 5 or 6 months now, so we're actually able to do all the stuff that we like. So, it would be actually funny if I say to you, I don't like this because it would mean that I didn't do my homework or didn't do it in a proper way.

Interviewer: How do you think developer perceive your team?

DvOps1: Emm.. I think they find it interesting because, you know what the funny part is… when we created this environment, we spoke with the whole team and management and I told them the first thing we have to do is onboard teams who are looking forward, who want to make move, who want to be on the cloud, because those are usually the people who have their applications ready for the cloud. They want to have their application ready for the cloud to the time to market, the time to deploy is rather fast because they want to. But if you have teams that are more in the legacy environment, it would be a big chunk of work and it would be annoying and would be you know, a lot of time or you know how all these things go, so It's very important to select the teams that are most willing, because they will actually help you in succeeding. So this is a very important part to me. To select the correct teams. So then, basically they're willing to join you and participate in this movement to the cloud.

Interviewer: Is there anything you think I have missed?

DvOps1: I don't think so… not that I can think of at the moment. I've explained how the process. There's also the fact that we have teams which are their….stationed in… for instance…now that's like 200km, so there's also this communication part sometimes, we do a lot through slack. Em…it's sometimes… it's most convenient or most ..if they're just sitting next to you. I experienced that could be particularly handy. And of course, there's good video connections and stuff…but I think the biggest ..x..is participating with the people. So, knowing who you're talking to, participating with them, and sitting together, looking at the application, try to help them. And when doing this, you, of course it takes time, you end up..the cooperation of the team, but eventually, you get the best result, I think. I think that's pretty clear to me as well. Whenever you sit together, you achieve the most. Let's put it that way. But also, understaning towards each other, to the teams, and the team also to the developers team, ops team, in this case, and so I think em.. I think that's something to point out in my experience. You need to sit together, that works. But other than that, I don't have much to add, I guess.

Interviewer: Is there anything else you would like to add? DvOps1: Not that I can think of. Interviewer: Would you mind If I come back to you with question later, maybe if I find I need to get clarification..

DvOps1: Sure. No problem, no problem..

Interviewer: ..in a few weeks or something

DvOps1: No problem, no problem..

Interviewer: Alright. And is there anyone else I can talk with about the study in your organisation, say a developer?

DvOps1: Em… I can certainly ask. The thing is that… because I am on the… on the pure infrastructure part of this process…. Other team members do, emm…. The development part. So, I'm not actually .. familiar on a good level with the developers but I can certainly ask.

Interviewer: Ok

DvOps1: And say I've done this interview.. about how DevOps team interact… is somebody willing to do a small interview? I can certainly ask. Interviewer: That would be great, thank you. Would you mind stating your name, where you work and how long you've been there? DvOps1: My name is DvOps1 XX, I am a DevOps Engineer, currently working at ING for the 5 months and I've been doing well IT well of course because we always change this naming convention from Ops to Unix to system admin to DevOps engineer. But I've been working in the IT industry professionally since I was 21 and I'm now 39 so that's 18 years.

Interviewer: Wow, this has been interesting. DvOps1: Thank you. Interviewer: I will switch off the recording now DvOps1: Ok

Appendix F

Participants' Description

F.1 Phase 1 participants

DvOps1

DvOps1 is a DevOps engineer with Finco1. DvOps1 has over 18 years' experience in software development and had been practicing DevOps for over 4 years. The participant has vast experience in Unix and Linux and has worked both as an engineer (maintaining enterprise Unix and Linux systems), and a consultant for both systems. DvOps1 is also a seasoned system architect. DvOps1 is an automation expert and has consulted with organisations looking to implement technologies like Kubernetes, AWS, Prometheus etc.

DvOps2

DvOps2 is a DevOps engineer with Finco2. DvOps2 has worked in the financial industry for over 12, as an IT operation expert. The participant is an agile leader in the organisation and a highly skilled Linux system administrator. DvOp2 has expert level experience and is certified in technologies like terraform, containerisation, object-oriented programming, and Kubernetes implementation. DvOps2 has a strong development background and has worked as a web developer and software engineer, before transitioning to IT operations and DevOps.

DvOps3

DvOps3 is a DevOps engineer with Finco1. DvOps3 has been in the software development industry for over 20 years. The participant is an AIX expert and highly skilled system administrator. DvOps3 has expert level experience in virtualization and storage area network and has been a Unix consultant.

DvOps4

DvOps4 is a Senior DevOps engineer with Finco3. DvOps4 has worked 7 years in the software development industry, 3 of which has been spent actively in DevOps-inclined activities. DvOps4 led the DevOps movement in the organisation and spearheaded the implementation of technologies Kubernetes, Terraform and AWS. DvOps4 has a background in web development.

DvOps5

DvOps5 is a Network DevOps engineer with Finco6. DvOps5 is an expert level practitioner in the software development industry, with over thirty-one (31) years' experience. DvOps5 has over twenty (20) years' experience working as a network data and security architect. The participant is an expert in private/public cloud technologies, as well as in network segmentation architectures. DvOps5 is a DevSecOps advocate and a DevOps professional.

DvOps6

DvOps6 is a DevOps engineer with ITCo1. DvOps6 has been in the software development industry for about 9 years, more than 6 of which has been spent in cloud infrastructure architecture and IT operations. DvOps6 is an AWS associate and serving at the technical team lead for the organisation, at the time of the interview. DvOps6 has a background in software development and is a certified project manager.

Dvr1

Dvr1 is a software developer with FinCo4. Dvr1 is an expert agile project manager, working in the financial industry for about 7 years (3 of which is in a DevOps-aware environment). Prior to this, Drv1 has worked as an enterprise software developer and an embedded systems consultant. Dvr1 is highly skilled in Oracle DB and serves as scrum master for his team.

Dvr2

Dvr2 is a software engineer and a full stack web developer with FinCo5. Dvr1 has been in the software development industry for about 3 years, all of which involved DevOps practices. The participant is familiar with the implementation of continuous integration and continuous delivery pipelines and participated in the delivery of an integrated software development platform in his organisation.

Dvr3

Dvr3 is a software developer at RegCo1. Dvr3 has over 15 years' experience of working in the software development industry and has spent 2 years in a DevOps-aware environment. The participant has a background in IT infrastructure maintenance and administration. Dvr3 is highly skilled in network administration and is an expert in cloud computing.

F.2 Phase 2 participants

DevCo1_CTO

DevCo1_CTO is the Chief Technology Officer of DevCo1, a software development organisation. DevCo1_CTO is a forefront DevOps thought leader. The participant has spent over 31 years in the industry, 9 of which is in DevOps. DevCo1_CTO is an expert level professional in Devops and cloud adoption and was actively involved in the transition to DevOps in his organisation. The participant has also consulted for other organisations on DevOps.

FinCo7_Lead

FinCo7_Lead is a senior DevOps Engineer in FinCo7. FinCo7_Lead has practiced in the software development industry for over 13 years, 8 of which has been in DevOps. FinCo7_Lead the lead architectural decision maker and directly engineered the design and building of a platform to support over 500 developers in the organisation. The participant has a background in system architecture and has led several DevOps initiatives.

MultCo1_DOps

MultCo1_DOps is the Engineering Manager in MultCo1. The participant has spent 12 years working in software development industry. MultCo1_DOps has been directly involved in DevOps implementation and practices for last 5 years. MultCo1_DOps is an expert in cloud engineering and Linux system administration. The participant has a background in IT infrastructure engineering and management

DevCo2_MD

DevCo2_MD is the Principal DevOps Consultant for DevCo2. DevCo2_MD is an IT professional with over 32 (thirty-two) years in the industry, and 9 years as a DevOps and continuous delivery consultant. DevCo2_MD has worked in several organisations as head of software development, chief software architect, and occupied other software development leadership positions. DevCo2_MD is a well-known author in the field.

DevCo3_Lead

DevCo3_Lead is the Principal Architect for DevCo3. The participant is a highly experienced senior technical manager and an expert in the delivery of automation solutions. DevCo3_Lead has spent over 17 (seventeen) years in the software development industry. DevCo3_Lead is responsible for the development, implementation of DevOps practices and cloud management platform in the organisation. DevCo3_Lead has also led DevOps adoptions in some other organisations.

DevCo4_CEO

DevCo4_CEO is the Principal Consultant for DevOps implementation in DevCo4. The participant has about 33 (thirty-three) years' experience in the software development industry, 11 (eleven) of which they have been involved in DevOps implementation and practices. DevCo4_CEO is a well-known author in DevOps, SRE, QA, and DevSecOps, and an expert in continuous integration and continuous delivery.

DevCo5_Lead

DevCo5_Lead is the lead DevOps engineer for DevCo5. DevCo5_Lead has spent over 20 (twenty) years in the software development industry, 9 of which is in the implementation of DevOps practices. DevCo5_Lead is an expert in site reliability engineering has experience across system administration and enterprise service management. The participant is also experienced in network and cloud engineering.

FinCo8_TM

FinCo8_TM is the Technical Manager for FinCo8. The participant has about 29 (twenty-nine) years' experience in the software development industry, and has been directly involved in DevOps implementation and practices for the last 9 years. FinCo8_TM is highly skilled across SD core areas such security, telephony, client/server, and database administration.

FinCo9_Lead

FinCo9_Lead is the Global DevOps Lead for FinCo9. FinCo9_Lead has 17 (seventeen) years' of software industry experience in delivery, consulting and management of IT products and services, 8 of which FinCo9_Lead worked as a software engineer and DevOps expert. The participant serves is responsible for the implementation of DevOps as well as continuous integration and continuous delivery (CI/CD) in the organisation.

DevCo6_DOps

DevCo6_DOps is a Senior DevOps Engineer in DevCo6. DevCo6_DOps is an expert in big data and database technologies and has been involved in numerous digital transformations. The participant has over 15 (fifteen) years of IT experience in high performance computing, computer architecture, cloud migration and software product delivery. DevCo6_DOps has been actively involved in DevOps implementation and continuous improvement for over 5 years.

ITCo1_Dvr

ITCo1_Dvr is a software Developer with ITCo1. ITCo1_Dvr participated actively in the DevOps implementation in the organisation. The participant has about 8 years' experience software testing, web application and software development.

DevCo8_Lead

DevCo8_Lead is the Lead Site Reliability Engineer in DevCo8. The participant has over 12 years' experience in software development and delivery. DevCo8_Lead is directly involved in the delivery of DevOps in DevCo8 and is contributing to continuous improvement of the software delivery practices in the organisation.

DevCo9_Chief

DevCo9_Chief is the Chief DevOps consultant for DevCo9. The main responsibility of the participant is to help the organisation successfully implement DevOps. At the time of the interview, the organisation had implemented the concept and making continuous improvement. DevCo9_Chief is a DevOps coach and an awardwinning speaker on the subject. DevCo9_Chief has been involved in several digital transformations, with 9 years' experience in DevOps implementation.

DevCo10_Chief

DevCo10_Chief is the Chief DevOps Consultant of DevCo10. The participant over 24(twenty-four) years' experience in the software development industry, performing several roles. DevCo10_Chief is a thought leader in continuous delivery and technology team organisation for fast flow and agility. DevCo10_Chief is an expert the delivery of online services, IoT, and embedded systems. DevCo10_Chief has a background in network engineering, software engineering and architecture.

DevCo11_Head

DevCo11_Head is the Head of Software Development for DevCo11. DevCo11_Head has over 22 years' experience in IT product and service delivery. The participant is an expert in large-scale software platform development, SRE, continuous delivery, and DevOps. DevCo11_Head has led several agile transformations and is responsible for the DevOps implementation in DevCo11.

ITCo1_DvOps2

ITCo1_DvOps2 is a DevOps Engineer with ITCo1. The participant was directly involved with the DevOps implementation of the organisation. ITCo1_DvOps2 has about 6 years' experience in DevOps. ITCo1_DvOps2 is highly skilled in technologies like containerisation, object-oriented programming, and Kubernetes implementation.

DevCo7_Eng

DevCo7_Eng is a DevOps Engineer with DevCo7. The participant is responsible for the implementation of DevOps practices and strategies for continuous integration/continuous delivery of Product Teams on Cloud Platform, in the organisation. DevCo_Eng is highly skilled in cloud migration, Azure, AWS, and automation.

DevCo7_Eng has a background in IT infrastructure management and Linux system administration.