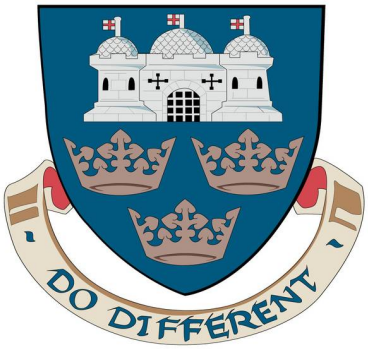
Changing Software Development Practice: A Case Study of DevOps Adoption

Stephen John Jones

A thesis submitted for the degree of

Doctor of Philosophy

University of East Anglia
Norwich Business School



October 2020

Abstract

DevOps, a portmanteau of development and operations, is a Software Engineering approach to emerge in industry, with a goal to rapidly develop and deploy good quality software. It has seen increased research attention in recent years with most studies focusing exclusively on tools used for DevOps or attempts to universally define it. This has led to a misunderstanding of DevOps alongside differing definitions, and therefore this research argues that a universal definition should not be sought.

A focus group of practitioners evaluated existing definitions with the findings further tested in a questionnaire to the wider DevOps community. The output of this informed a 14 month case study of DevOps adoption in a medium sized UK organisation. A pragmatic approach was taken to study what DevOps meant for the organisation and its impact on employees and other business functions.

This research contributes to theory by identifying the core attributes of DevOps, and by using a job crafting theoretical lens to understand the organisational change required to implement DevOps and elucidating how individuals change their work identity as they adopt DevOps practices and processes. In particular, this research finds that Software Developers are natural Job Crafters, especially if afforded the freedom to do so. This research contributes methodologically by using multiple methods, and in particular a longitudinal qualitative diary study over 14 months with a very low attrition rate. This was achieved through using tools that participants use in their work to record their experiences of DevOps implementation. Finally, this research makes a practical contribution by developing the building blocks of attributes that organisations should consider within their specific context and by developing an interdisciplinary framework that takes account of both the software development process and the associated management implications of adopting and implementing DevOps.

Contents

List of Figures

List of Tables

Dedication

Dedicated to and in memory of my beloved son, Micah George Jones, who was born sleeping on 04 Oct 2015 at 19:15. He was, and still is, the driving force of my will to succeed in all things I undertake.

The world may never notice
If a Snowdrop doesn't bloom,
Or even pause to wonder
If the petals fall too soon.
But every life that ever forms,
Or ever comes to be,
Touches the world in some small way
For all eternity.

The little one we long for
Was swiftly here and gone.
But the love that was then planted
Is a light that still shines on.
And though our arms are empty,
Our hearts know what to do.
Every beating of our hearts
Says that we love you.

- author: unknown.

Love bears all things, believes all things,
hopes all things, endures all things.
- 1 Corinthians 13:7 (ESV)

Acknowledgements

This work was the most challenging intellectual undertaking of my life. It was filled with challenges including a life changing road traffic accident in June 2018, which almost cost me my life let alone my PhD when I was in the final stages of writing this thesis. I would like to dedicate this small section to a some outstanding individuals and organisations, the support of whom has been invaluable.

Mrs. Claire Jones
Your continued support and love for me means so much. You have been through so much and nearly lost me in 2018. You are as much my best friend as you are my wife, I hope this work makes you proud. I look forward with optimism to the next big chapter in our lives and marriage.

Professor Fiona Lettice (UEA) and **Dr Joost Noppen (BT)**
Without your guidance or support throughout this long and eventful journey, there is no way I would have finished my research. I view you both as mentors, let alone supervisors. You have enthused me with research and encouraged me to explore it further. I have fallen in love with Business Management and Software Engineering as research disciplines. I have enjoyed learning from you both and discovering the researcher in me.

Dr David Cutting, **Dr Sultan Al-Khatib** and **Mr Adam Ziolkowski**
We had some great times over the years studying under Joost's supervision and without your fellowship and humour, it would not have been the same. Thank you guys for your support and the occasional (frequent in Dave's case) insult and verbal abuse. In the end and despite our best efforts, we all failed to disappoint Joost; although Dave, you did come close!

Professor Ana Sanz Vergel (UEA)
I am very grateful for your constant encouragement with my research. I remain greatly inspired by you following your keynote talk in the Norwich Business School doctoral colloquium. Thank you very much for your continued support and inspiration throughout! Me gustaria en esta ocasión agradecerle su interes, apoyo y ayuda a lo largo de mi trabajo.

Norwich Business School and **School of Computing Sciences**
It has been a privilege to study and undertake teaching within both schools. Thank you for providing the facilities that greatly helped provide a conducive environment for undertaking this work.

Anglia Farmers Ltd.
Thank you for offering me the opportunity and privilege to work with you to undertake my research and to present the findings.

East Anglian Air Ambulance and **East of England Ambulance**
Without your timely intervention in June 2018, I would not be in this position now. You not only saved my life, but also my greatest academic achievement. Words cannot even begin to express my gratitude.

Addenbrooke's, Papworth and Colman Hospital
You helped me through one of the darkest periods of my life. While the recovery will take years, I am grateful for the amazing care and rehabilitation therapy I have received from you.

Mr Dom Davis, **Mr Jason Gibbs** and **Tech Marionette Ltd.**
Thank you so much for providing me a desk in your office and for the conducive environment to help me manage my fatigue levels, work on this thesis as well allow me to explore and reinvigorate my technical skills and re-engage in the Norwich tech network. Taken together, this contributed not only to my ongoing recovery but also the completion of this work.

SOUL Church
Thank you for being supportive, accommodating and for being able to make use of your cafe to complete writing tasks as well as to re-explore my faith. I have become fond of your amazing coffee and home made lunches.

Almighty God
You have been my strength throughout this entire process and have been a source of comfort in those times I felt hopeless. It is my great desire that this thesis helps to advance the knowledge of DevOps and brings you glory.

List of Papers

Published:

Jones, S., Noppen, J., and Lettice, F. (2016). Management Challenges for DevOps Adoption within UK SMEs. In *Proceedings of the 2nd International Workshop on Quality-Aware DevOps, July 21 2016, Saarbrücken, Germany*, pages 7–11. ACM

In Progress:

Jones, S., Zimpel-Leal, K., Lettice, F. and Noppen, J. DevOps: A Framework for Contextual Definition.

Jones, S., Lettice, F., Noppen, J. and Zimpel-Leal, K. Changing Software Development Practice: What DevOps Means For Organisations.

Jones, S., Noppen, J., Lettice, F., Davis, D. The DevOps Particular: Comparing Practice in a Startup, SME and Large Business.

Jones, S., Sanz Vergel, A., Lettice, F. and Noppen, J. Job Crafting in Software Engineering: A Qualitative Study.

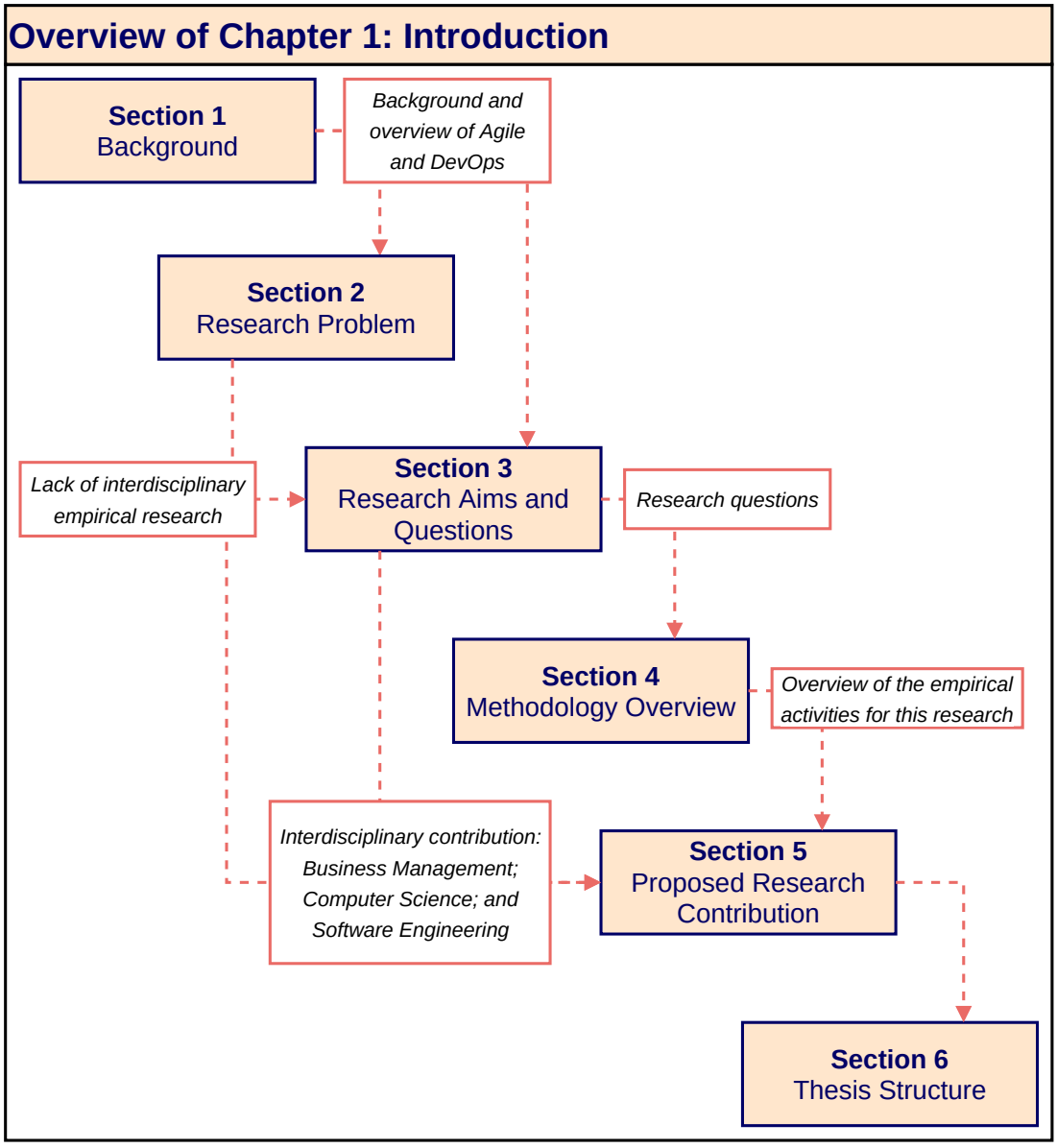
Jones, S., Lettice, F. and Noppen, J. Peeling the Research Onion to map and guide longitudinal studies in Software Engineering.

Chapter 1

Introduction

"DevOps shows how we optimize the IT value stream, converting business needs into capabilities and services that provide value for our customers"

– Gene Kim


Overview of Chapter 1: Introduction
Section 1
Background
Background and overview of Agile and DevOps
Section 2
Research Problem
Lack of interdisciplinary empirical research
Section 3
Research Aims and Questions
Research questions
Section 4
Methodology Overview
Overview of the empirical activities for this research
Interdisciplinary contribution: Business Management; Computer Science; and Software Engineering
Section 5
Proposed Research Contribution
Section 6
Thesis Structure

1.1 Background

Software development methods tend to focus solely on the software development teams. One significant development in the late 20th century was the Agile approach to software development, which advocates team empowerment through an iterative and incremental approach, and tackled many of the issues around slow, linear or siloed approaches encountered with early software development [Šmite et al., 2010]. In the early 21st century, the manifesto for Agile software development was published [Beck et al., 2001], with focus placed on four core values:

Individuals and Interactions over processes and tools
Working Software over comprehensive documentation
Customer Collaboration over contract negotiation
Responding to Change over following a plan

However, once software is developed within an organisation, it is typically passed to an Information Technology (IT) operations or support team, who become responsible for its deployment, ongoing maintenance and provision of support. This approach still leads to organisational silos, introducing further socio-cultural and socio-technical issues between both functions, such as a blame culture [Hussaini, 2014; Loukides, 2012; Mohamed, 2015; Tseitlin, 2013], communication difficulties [Bass et al., 2013; Hussaini, 2014] and delays in producing and deploying software updates [Chen, 2015; Hussaini, 2014].

Industry is increasingly moving towards the integration of both software development and IT operations functions to avoid some of these problems. This integration is core to DevOps, which places a major emphasis on high levels of collaboration between software development and IT operations functions [Cook et al., 2012; Erich et al., 2014; Hussaini, 2014; Kantsev, 2017; Loukides, 2012] (see figure 1.1).

DevOps is a portmanteau of Development and Operations, which originated within web development organisations as a response to satisfying increasing de-

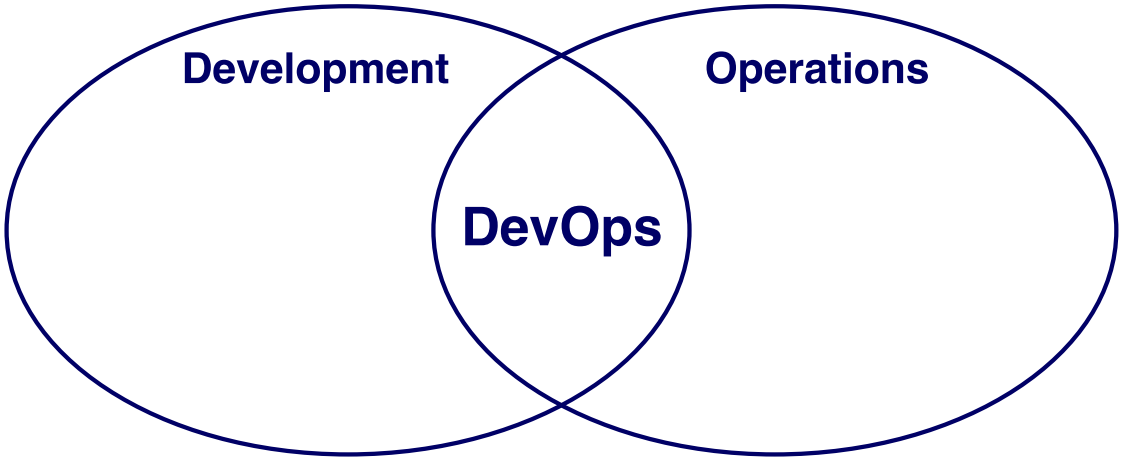


Figure 1.1: DevOps is often represented as a Venn diagram [Kantsev, 2017]

mand for the rapid development and deployment of good quality software [Liu et al., 2014]. Furthermore, it is argued that the functional integration of software development and IT operations can potentially bring big benefits to organisations from the rapid delivery of new features and updates to existing software through improvements to business IT infrastructure and process harmonisation [Claps et al., 2015; Liu et al., 2014].

1.2 Research Problem

Existing contributions to the DevOps knowledge are mostly focused on technical aspects specifically with the various tools associated to it. As such, much of the literature appears positioned within the Computer Science academic discipline. There is also an overall lack of empirical research on what DevOps adoption means for an organisation.

However, one of the largest issues with the published literature is how DevOps can be defined. Although there is no shortage of definitions, DevOps appears difficult to define [Smeds et al., 2015]. Loukides [2012] is cited by many authors in their attempts to define DevOps, but despite the detail offered, fails to provide a single definition [Smeds et al., 2015]. Dyck et al. [2015] argue that existing

definitions for DevOps lack clarity and do not especially distinguish it from other software engineering approaches.

Dyck et al. [2015] also argue that many definitions fail to distinguish DevOps from release engineering. Dyck et al. [2015, 3] define release engineering as "a software engineering discipline concerned with the development, implementation and improvement of processes to deploy high-quality software reliably and predictably". The key point that Dyck et al. [2015] argue, is that a release engineer can only function to this definition if, and only if, decent communication and information flows exist between all involved with the development, deployment and maintenance of software and infrastructure.

Dyck et al. [2015, 3] therefore attempt to define DevOps universally as an "organizational approach that stresses empathy and cross-functional collaboration within and between teams - especially development and IT operations - in software development organizations, in order to operate resilient systems and accelerate delivery of changes."

The notion of poor collaboration between development and IT operations is also picked up by Hosono [2012, 330], where DevOps is defined as a "practice aimed at repairing the schism between the two teams". It can only be assumed that Hosono [2012] is referring to software development and IT operations teams. Moreover, this definition postulates that both functions exist as silos, and that perpetual conflict exists between them (see figure 1.2).

Nevertheless, it is common knowledge that organisational silos can lead to conflict and schisms [Diamond et al., 2004; Tseitlin, 2013]. However, Hosono [2012] does not provide any evidence of a significant schism between software development and IT operations, and where DevOps is solely aimed at closing such schisms.

This thesis explores several definitions found in the literature, but they mostly appear secondary, derived from non-peer reviewed sources, such as web pages and blogs, rather than by empirical means or peer-reviewed sources. Additionally, and critically, a lack of methodological transparency and rigour compounds the issue. For instance, Dyck et al. [2015] offer very little transparency of the process leading

Figure 1.2: A meme used to comically illustrate blame passing between software development and IT operations [Roche, 2013, 39].

to their universal definition, or the level of rigour they employed.

Furthermore, some definitions appear inconsistent and conflicted. For instance Hosono [2012] focuses on conflict, whereas Császár et al. [2013] stress that DevOps is about practice and performance. A list of definitions discovered in the literature are presented in Chapter 3 (see table 3.1), and taken together, the definitions presented illustrate the difficulty in defining DevOps.

1.3 Research Aims and Questions

Given that organisations are increasingly reliant on software within their operations and that DevOps is concerned with the development and deployment of

software systems, the primary aim of this thesis is to understand what DevOps is, how it is adopted by organisations and what it means for them. This thesis positions DevOps as an interdisciplinary topic covering Business Management, Computer Science and Software Engineering.

This research aims firstly, to identify and present a review of DevOps research from academic and industrial sources. Secondly, to identify core attributes of DevOps and provide a framework which can be used to help develop a definition, or validate an existing one. Finally, this research seeks to pragmatically identify and explore the business management challenges associated with adopting DevOps by means of a longitudinal case study of a UK organisation adopting DevOps. The following questions are posed to drive this research:

1. How can DevOps be defined?

2. Why do organisations adopt DevOps?

- **2a.** What are the perceived performance or strategic benefits?
- **2b.** How is DevOps different to other approaches for software development?
- **2c.** Are anticipated performance gains from its implementation realised?

3. How does DevOps adoption influence software development processes?

- **3a.** What changes are required to the organisation and management of software development processes to enable DevOps?

4. How do software development and IT operations roles, tasks, skills, tools and work identity change as DevOps is adopted within an organisation?

1.4 Methodology Overview

This PhD research is divided into two distinct phases. The first phase explores the definition of DevOps, taking a mixed methods approach, including a focus group

of DevOps practitioners, and a questionnaire to the wider DevOps practitioner community. The focus group considers existing definitions already present in the literature, but also seeks to identify and investigate attributes any DevOps definition should consider. The questionnaire is used to validate the focus group's output and help to refine a thematic analysis used in the second phase of this research.

The second and main phase, is a longitudinal case study over a 14 month period to explore the actual adoption of DevOps within a medium-sized UK based business, with a particular focus on the management challenges and implications. This is accomplished through an inductive multi-method qualitative approach using an open reflection diary study and semi-structured interviews with managers, software developers and IT systems administrators.

1.5 Research Contribution

This thesis bridges the disciplinary gap between Business Management, Computer Science and Software Engineering in the study of DevOps. Firstly, the thesis provides a literature review of relevant literature and identifying key overlapping themes.

The literature lacks any longitudinal studies of DevOps adoption. This thesis provides a case study of a medium sized UK business adopting DevOps for the development and deployment of a business critical software system. Defining DevOps has been problematic since it emerged as a topic. This overlaps with Agile when it too first emerged. This thesis does not offer a universal definition for DevOps. However, it does provide both management and technical attributes that any definition should consider.

The literature also postulates DevOps as being a harmonisation of two functions, software development and IT operations. Yet this thesis provides a second possibility of DevOps realisation where no total harmonisation occurs. From the case study, DevOps is achieved by the software developers through drastic changes

in team dynamic, work identity and function. This in turn was driven by job crafting, predominately observed with software developers to a point where individuals changed the way they identify at work. This research is the first to show the role that job crafting plays in the adoption of DevOps in a software development context, therefore offering a further contribution to the job crafting literature.

1.6 Thesis Structure

This thesis is presented over a number of chapters as illustrated in figure 1.3, which also shows the inputs and outputs for each chapter.

Chapter one begins the thesis by introducing DevOps and sets out the agenda of this PhD research. Chapter two details the research methodology and precedes the literature review in order to provide the methodological approach and context for the systematic literature review presented in Section 3.3.

Additionally, chapter two discusses abductive reasoning around job crafting following the piloting of diaries and interviews used for the case study presented in chapter five. A review of literature is provided in chapter three, which includes a narrative overview of software, its origins and why it is important in business management research. A systematic review of the DevOps literature is then provided followed by an overview of job crafting theory.

Chapter four presents the findings from a focus group and questionnaire survey regarding the definition of DevOps. Chapter five puts forward an overview of themes derived from qualitative data collected over 14 months when following Anglia Farmers Ltd.'s adoption of DevOps, of which a case study is also presented. Chapter six discusses the findings of this research and synthesises them with the literature before offering a conclusion, outlining the theoretical and methodological contributions this research makes. Practical recommendations are provided alongside acknowledged limitations of this research before directions about future interdisciplinary work on DevOps are put forward.

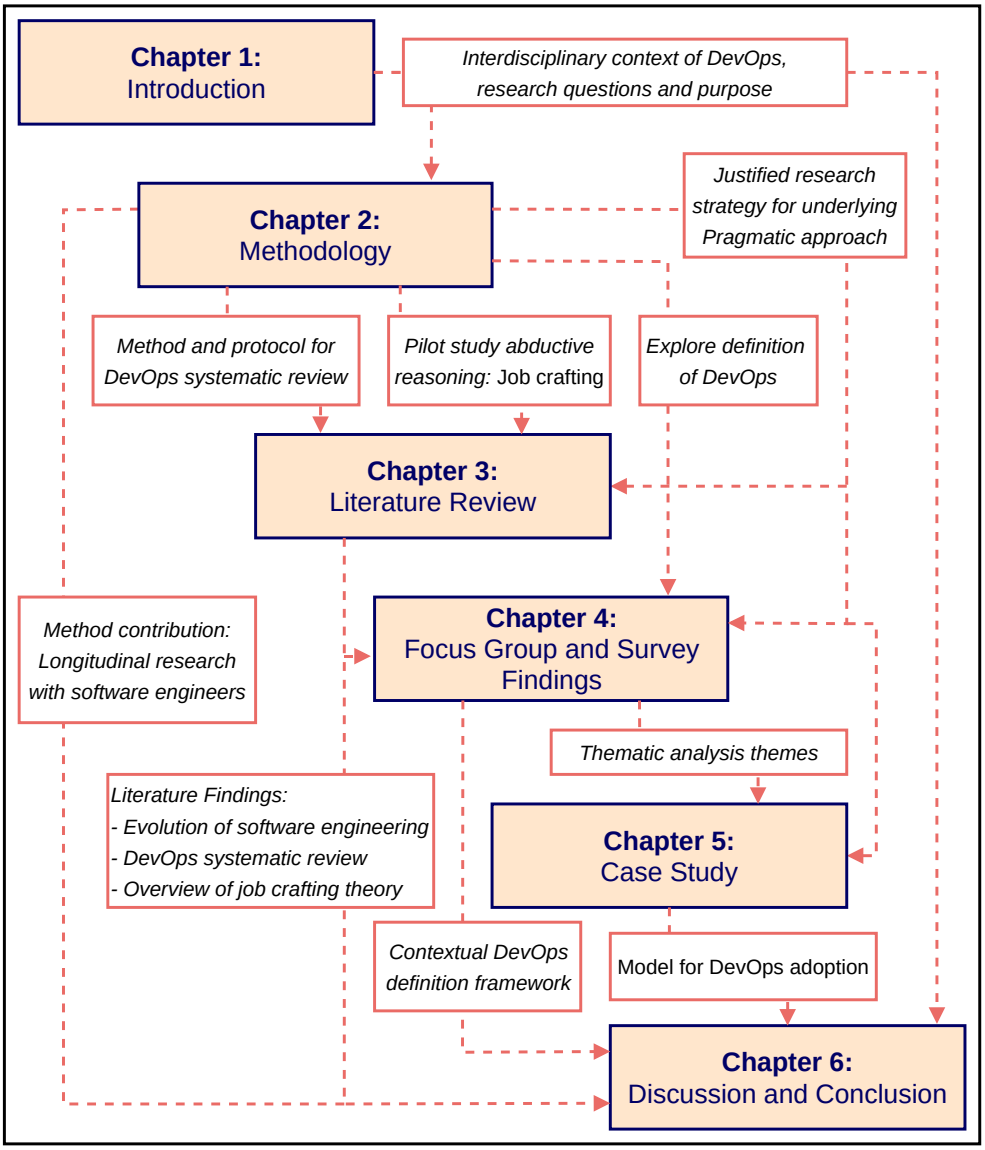


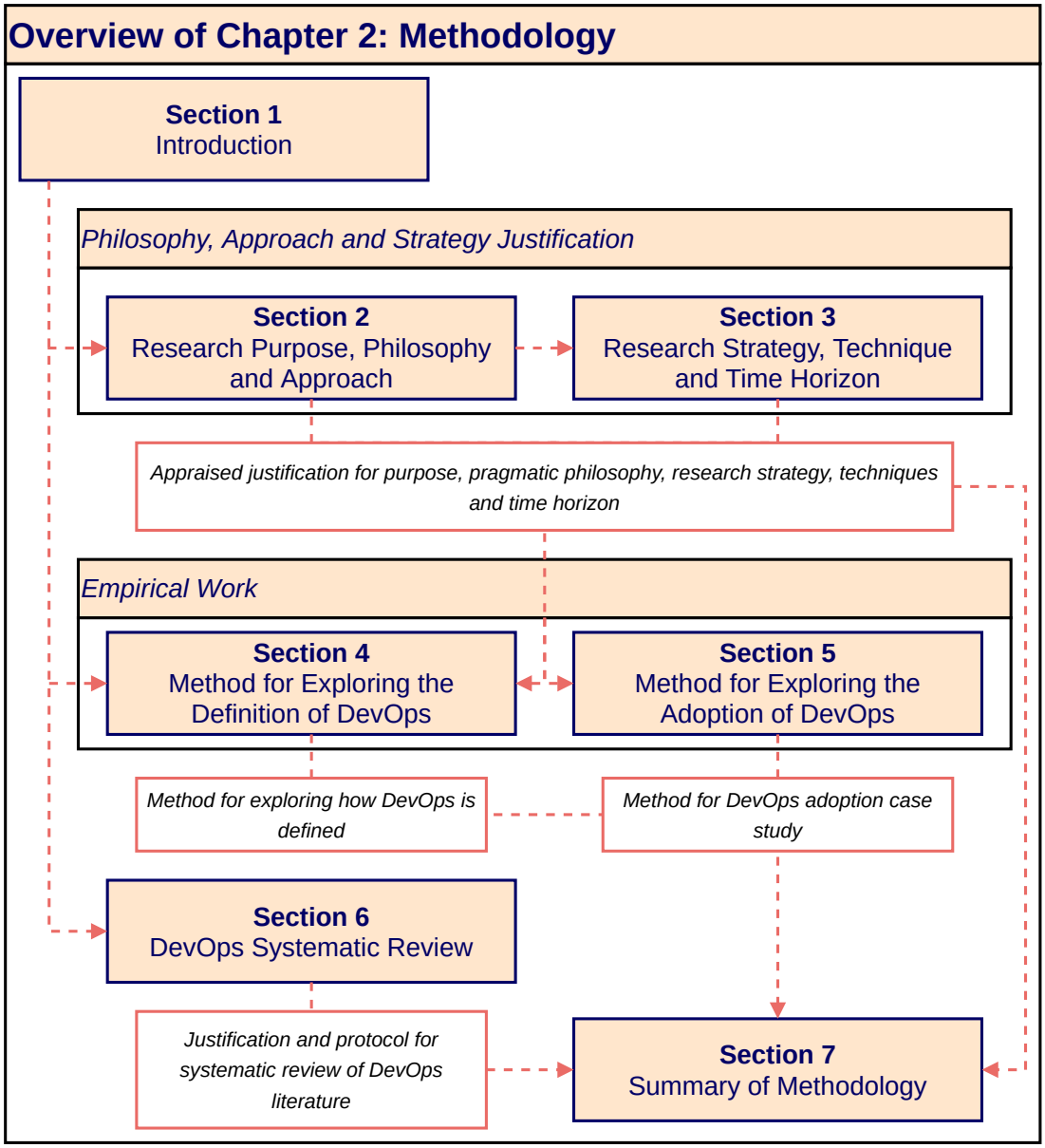
Figure 1.3: Structure of this thesis with chapter inputs and outputs.

Chapter 2

Methodology

"We are trying to prove ourselves wrong as quickly as possible, because only in that way can we find progress."

– Richard Feynman


Overview of Chapter 2: Methodology
Section 1
Introduction
Philosophy, Approach and Strategy Justification
Section 2
Research Purpose, Philosophy and Approach
Section 3
Research Strategy, Technique and Time Horizon
Appraised justification for purpose, pragmatic philosophy, research strategy, techniques and time horizon
Empirical Work
Section 4
Method for Exploring the Definition of DevOps
Section 5
Method for Exploring the Adoption of DevOps
Method for exploring how DevOps is defined
Method for DevOps adoption case study
Section 6
DevOps Systematic Review
Justification and protocol for systematic review of DevOps literature
Section 7
Summary of Methodology

2.1 Introduction

This chapter provides a detailed discussion of the methodology for this research, including philosophical considerations, principles, procedures and processes, all of which form a guideline to studying DevOps, within an industrial context. In addition, the methodological choices are appraised and discussed, to address the research questions.

In particular, this chapter firstly discusses the research considerations, justifying why a pragmatic stance is taken, what the purpose of the research is and the subsequent strategy employed. Fieldwork instruments are introduced, including a mixed method approach of a focus group and questionnaire designed to explore the definition of DevOps. This helps to fine tune a qualitative diary study and semi-structured interviews for a 14 month case study exploring DevOps adoption at Anglia Farmers Ltd. (AF)[1]. The chapter then provides details about the systematic component of the literature review as well as the protocol for it.

2.2 Research Purpose, Philosophy and Approach

Saunders et al. [2011, 42] argue the importance of clear purpose by the metaphorical position of "contracting with your client" in the manner that it would simply be unthinkable to carry out any research in such a manner without any clear purpose or proposal. With the research philosophical stance and subsequent approaches established, the next item to discuss is the overarching purpose for the research, which in addition to making a contribution, may be "to explore, to describe and/or to explain" [Robson and McCartan, 2016, 39]. Before stating the purpose for this research, each position is briefly appraised.

Exploratory research seeks new insights into phenomena and discovering what is happening and why, through the asking of questions and appraising phenomena in a new light [Saunders et al., 2011]. While such inquiry can be qualitative or

[1] http://www.angliafarmers.co.uk/ accessed: June 2017

quantitative in nature, the former approach is generally favoured given the focus on new areas of research [Robson and McCartan, 2016].

Descriptive research focuses on the accurate representation of events, persons and/or situations. It can be either qualitative or quantitative in nature but requires comprehensive previous knowledge of the subject [Zikmund et al., 2013]. Descriptive research can therefore contribute much greater insights on existing topics [Robson and McCartan, 2016].

The eponymous explanatory research focuses on the subject, seeking to explain any relationships between it and any other variables [Saunders et al., 2011]. Explanatory research can be qualitative or quantitative in nature.

The scoping of the DevOps topic continually influenced the research purpose by pushing it further down an exploratory road. DevOps is a relatively young topic in industry, but also in academic research where it is still new in Computer Science research, and is largely untouched in the Business Management discipline.

This section will discuss the philosophical considerations and research approach. The influence of philosophy and its subsequent views for the entire research project is illustrated in figure 2.1, and forms the basis for the structure of this section.

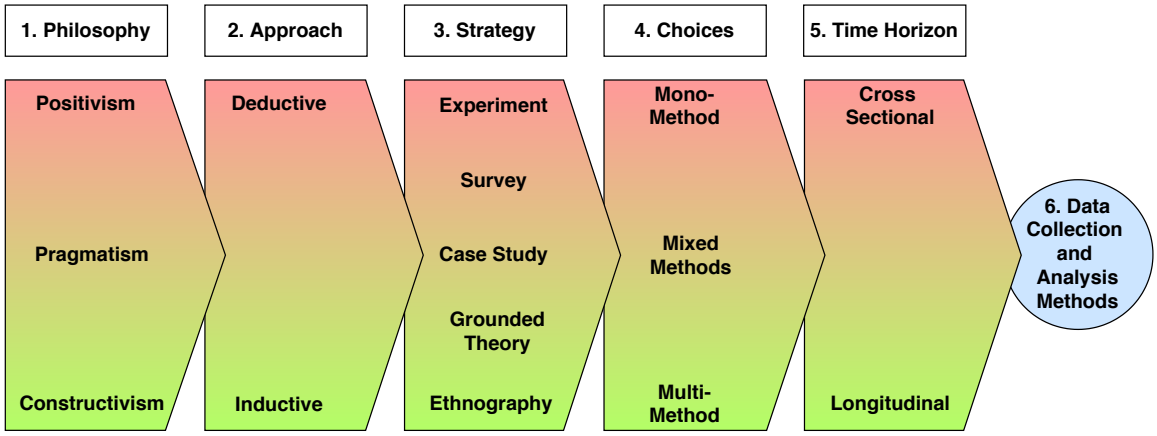


Figure 2.1: Non-exhaustive flow of research considerations derived from Gill and Johnson [2010]; Robson and McCartan [2016]; Saunders et al. [2011].

2.2.1 Research Philosophy

It is prudent to begin by discussing the philosophical debates present in the Social Sciences and the "mutually exclusive relationship between method and philosophy" [Knox, 2004, 119]. As such, concerns over research strategy and data collection methods are secondary in nature to those of the underlying philosophical stance [Guba et al., 1994]. This section will briefly discuss the ontological and epistemological concerns of Positivism, Constructivism and Pragmatism.

2.2.2 Evaluation of Positivism, Constructivism and Pragmatism

The philosophical views of Positivism and Constructivism have opposing ontology, or views on reality and epistemology, or what is considered acceptable knowledge of said reality [Robson and McCartan, 2016; Saunders et al., 2011]. Pragmatism is a third philosophical view which can accept both the views of Positivism and Constructivism where such views allow the job to be done. Therefore, Pragmatism offers a potential philosophical middle ground [Robson and McCartan, 2016].

Positivism asserts that reality is objective and exists externally to the individual, meaning it can be measured through scientific method [Saunders et al., 2011]. As such, empiricism is commonplace with a reliance on quantitative methods within the Positivist view [Robson and McCartan, 2016]. Although coupled with scientific method, Positivists take the view that all collected data and observations must be objective and have no influence from the researcher, making a further assertion that science is value-free. Positivist research therefore typically contributes to theory building through the deductive approach [Saunders et al., 2011].

On the other hand, and in contrast, Constructivism is an interpretive view asserting that the nature of reality is mentally constructed by an individual's experience and knowledge through cognitive and social interaction processes [Saunders et al.,

2011; Young and Collin, 2004]. Thus, Constructivist researchers generally struggle with and reject any notion of an objective reality [Robson and McCartan, 2016]. Favouring the collection of data from social interactions and observations in a natural context, Constructivism seeks to understand what is happening and why. Therefore, Constructivists employ strategies that favour qualitative enquiry, including case studies, ethnography and interviews.

Seeking a "middle ground between philosophical dogmatism and scepticism", Pragmatism strongly advocates there to be 'no one correct way' [Robson and McCartan, 2016, 29]. Pragmatism therefore asserts that the single most important determinant of any ontology or epistemology one adopts falls around the research question itself [Robson and McCartan, 2016; Saunders et al., 2011]. As such, Pragmatists base methodological decisions around what they feel is compatible with their own values, which are derived culturally, which is the reality for many researchers investigating social and behavioural topics [Teddlie, 2005].

For the Pragmatist researcher, it means that, for instance, a deductive approach could be taken for one research question, whereas an inductive approach is taken for another. Table 2.1 provides a summary breakdown of the key differences between Positivism, Constructivism and Pragmatism. The first of these differences is the research approach, which further underpins the strategy and methods utilised within any research.

A deductive approach involves the testing of theoretical propositions which the research strategy and accompanying methods are specifically designed for [Saunders et al., 2011]. Typically, hypothesis testing and large samples of quantitative data are favoured, with outcomes examined and the theoretical proposition in question accepted, rejected and/or revised [Robson and McCartan, 2016].

	Positivism	Constructivism	Pragmatism
Approach	- Deductive	- Inductive	- Deductive and/or Inductive
Assertions	- Objective reality. - Researcher independence. - Science is value free. - Causal analysis leading to law-like generalisations.	- Subjective reality which is socially constructed. - Researcher part of observation. - Empathetic stance to science. - Unique and subjective understanding held by participants.	- Objective and Subjective Realities. - Action over Philosophising - Advocates human experience. - Endorses fallibalism and provisional truths.
Strategies	- Controlled hypothesis formulation and testing - Rigid and highly structured - Emphasis on quantifiable observations	- Minimal research structure. - Develop studies through ongoing induction from the data. - Emphasis on observing phenomena in its natural setting.	- Eclectic and Pluralist. - Endorses practical empiricism to determine what works. - Value-oriented approach derived from culture.
Method Preference	- Generation of quantitative data through large samples - Rigour and validity - Generalisation	- Generation of qualitative data through smaller samples - Trustworthy interpretation and triangulation - Contextual understanding	- Generation of qualitative and/or quantitative data. - Human enquiry analogous to experimentation and scientific enquiry.

Table 2.1: Comparative overview of Positivist, Constructivist and Pragmatist philosophical views (adapted from Guba et al. [1994], Gill and Johnson [2010], Saunders et al. [2011] and Robson and McCartan [2016]).

On the contrary, an inductive approach is far more focused on developing theory as a result of the research activities [Saunders et al., 2011]. While hypothesis testing can still be undertaken within inductive research, it will typically come following data collection instead of before it. As such, inductive studies may refer to hypothesis generation, as opposed to hypothesis testing. Theory is developed, or informed following research activities [Robson and McCartan, 2016].

2.2.3 Philosophical Stance and Approach Taken

Saunders et al. [2011, 109] make the point that "which is better depends on the research questions". They continue, further adding that researchers would be deluding themselves if they believed their research questions fell perfectly into one specific philosophical domain; and this PhD research is no different.

This PhD research adopts a Pragmatic philosophical stance given the selection of appropriate positions in order to answer each research question [Robson and McCartan, 2016]. It therefore also assumes there is no one right or correct stance or approach to take [Saunders et al., 2011]. As discussed previously, pragmatism places the research questions at the centre when making decisions concerning the approach and subsequent strategy, method choices and time horizons. The research questions for this PhD project are defined below:

1. How can DevOps be defined?

2. Why do organisations adopt DevOps?

- **2a.** What are the perceived performance or strategic benefits?
- **2b.** How is DevOps different to other approaches for software development?
- **2c.** Are anticipated performance gains from its implementation realised?

3. How does DevOps adoption influence software development processes?

- **3a.** What changes are required to the organisation and management of software development processes to enable DevOps?

4. How do software development and IT operations roles, tasks, skills, tools and work identity change as DevOps is adopted within an organisation?

This research seeks answers to how DevOps is defined along with what would constitute as a core attribute of it. As this portion of the research is taking existing propositions and testing them, a Positivist deductive approach was taken in answering the first research question.

The most substantial portion of this research is the investigation of actual DevOps adoption within an organisation. For this, a Constructivist inductive approach was taken, placing premise on “the particular” [Stake, 1994, 38], in this case DevOps, and most importantly, the human experience within it. This approach is appropriate for the remaining research questions given the emphasis on studying the often diverse and differing perspectives of the participants [Stake, 1995, 2000] rather than the experimental study of tools used in such roles, a common theme within the existing academic and industrial DevOps literature.

2.3 Research Strategy, Technique and Time Horizon

This section appraises and discusses potential research strategies, data collection techniques and time horizon, and their methodological fit with the underlying research purpose, philosophy and approach which was outlined and discussed in section 2.2.

Saunders et al. [2011, 600] define the research strategy as the "general plan of how the researcher will go about answering the research question(s)". Thus the research strategy guides the researcher throughout the project, underpinning the choice and justification of data collection methods. In pragmatic research, the strategy links methods to the purpose and ultimately, the questions.

There are a range of research strategies, including experiment, survey, case study,

grounded theory and ethnography, which are briefly set out before those chosen for this PhD research are justified according to their fit with the purpose, philosophy, approach and research questions set out previously.

2.3.1 Experiment

The experiment research strategy, while well applied within social sciences, in particular Psychology, originates from the natural sciences, where the core purpose is the study of causality between observed variables [Saunders et al., 2011]. Typical experiment research within the social sciences involves grouping participants into two types of group. Firstly, an experimental or intervention group, whereby the variables under observation are manipulated, and, secondly, a control group, where they are not, thus enabling the difference to be observed and reported [Saunders et al., 2011].

As a strategy for exploratory and explanatory research purposes, experiments can provide answers to how and why questions. However, research activities are typically conducted under highly controlled and/or laboratory conditions. As a deductive strategy, hypothesis testing is typical with experiments, and thus the results, by virtue of the controlled conditions from which they were produced, are unlikely to bear much resemblance with the real world nor be feasible for many Businesses Management topics [Robson and McCartan, 2016; Saunders et al., 2011].

2.3.2 Surveys

Often associated with the deductive approach, the survey strategy is popular in business management research as it can answer the "who, what, where, how much and how many questions" and is therefore well suited for exploratory and descriptive research purposes [Saunders et al., 2011, 144]. Surveys allow for the collection of both qualitative and quantitative data, although the latter can be collected from large samples efficiently.

With collected quantitative data, analyses can be undertaken using descriptive and inferential statistical techniques [Saunders et al., 2011]. Combined with sampling, the survey strategy offers a degree of control over the research process, but not necessarily to the extent that experiments do, and can potentially produce results which are representative of a population through generalisation.

While enabling large amounts of data to be collected, the survey strategy can have limitations in the breadth of that data, as opposed to other strategies.

2.3.3 Case Studies

A case study can be defined as "an in-depth exploration from multiple perspectives of the complexity and uniqueness of a particular project, policy, institution, programme or system in a 'real life' context" [Simons, 2009, 21]. As a research strategy, the case study enjoys wide application across a variety of academic disciplines [Thomas, 2011], and is especially established within Business Management research [Welch et al., 2011], proving exceptionally popular with qualitative researchers [Piekkari et al., 2009]. As such, the case study makes for an excellent strategy for both exploratory and explanatory research [Saunders et al., 2011]. However, there is ongoing philosophical debate around case studies, which this section will attempt to summarise.

Thomas [2011, 512] offers a concise breakdown of any case study into two constituent parts, the "subject" and "object". The subject refers to the case itself [Thomas, 2011], the "phenomenon in its natural context" [Piekkari et al., 2009, 569] or "the particular" [Stake, 1994, 238]. The object refers to the context in which the subject is studied [Thomas, 2011], and therefore offering the "means of interpreting or placing" the subject in context [Thomas, 2011; Wieviorka, 1992].

However, the object or "boundary" is the subject of philosophical discussion [Piekkari et al., 2009, 572]. The rhetoric of this discussion revolves around Positivist arguments that the boundary should be set and fixed [Eisenhardt, 1989; Piekkari et al., 2009; Wieviorka, 1992; Yin, 2013] as opposed to Constructivists

who advocate a flexible approach depending on the observations of the subject [Piekkari et al., 2009; Stake, 1994; Thomas, 2011].

As a research strategy in Business Management research, the case study can be attributed to Kathleen Eisenhardt [1989] and Robert Yin [2013]. First published in 1984, Platt [1992, 44] applauds Yin's work as the "best known modern work" on the case study. Eisenhardt builds on Yin's work, but with a specific differences: Eisenhardt has an academic focus on theory building using a single case, while Yin utilises multiple cases in a more practical manner, aiding with policy making and consulting [Piekkari et al., 2009].

While the case study has become "an increasingly popular and relevant research strategy" [Eisenhardt and Graebner, 2007, 30], it is fraught with criticism especially around its application and the dominant Positivist philosophical undertones, which in turn influence research through data collection and analysis methods [Piekkari et al., 2009].

Piekkari et al. [2009] argue that the case study requires greater understanding, and more importantly, a fuller grasping of the disciplinary context within which the case study is being utilised. Furthermore, Piekkari et al. [2009], based around philosophical positioning, categorise the case study into three distinct categories: Positivist; interpretivist and critical realist. However, only in recent years have interpretevist and critical realist case studies emerged.

Above all, Piekkari et al. [2009] argue that these philosophical undertones, while seemingly arcane, do matter in case study research. For instance, case studies have been utilised to firstly, inductively develop theory, followed by using them again to deductively and empirically test the developed theory, thus completing a cycle [Eisenhardt and Graebner, 2007]. This usage of the case study merely "constitutes a means to an end" [Piekkari et al., 2009, 5], failing to consider any questions in the causality inherent of Positivist approaches to theorizing [Ragin, 1992, 1997].

Constructivists such as Stake [1994, 238] take a different view, arguing that the aim of the case should be the "study of the particular". This assertion of the case

study's aim involves "understanding of human experience" [Stake, 1995, 38], which Stake considers the main purpose of any theorizing which can be distinguished epistemologically, at least, from causal explanation [Piekkari et al., 2009].

The data used to construct a case study can be collected through a variety of methods. As such, it is necessary for the researcher to triangulate the data collection so as to "ensure the data are telling you what you think they are telling you" [Saunders et al., 2011, 146].

Grounded Theory

Grounded theory arose in 1967 from the work of Glaser and Strauss [2017], and has often been oversimplified in attempts to define it as "the best example of inductive research" [Robson and McCartan, 2016; Saunders et al., 2011, 148]. Instead, Saunders et al. [2011] argue grounded theory should be considered as a means to building theory by combining deductive and inductive approaches, which can be particularly well suited to both explaining and predicting behaviour [Robson and McCartan, 2016].

Put simply, grounded theory asserts that theory is generated from the data [Glaser and Strauss, 2017] and that its collection begins without a prior formation of any theoretical framework [Saunders et al., 2011]. It is argued that grounded theory is by no means theory testing and that in order to draw conclusions and theoretical insight, data should be collected at a conceptual level [Suddaby, 2006].

Indeed, Suddaby [2006] continues with drawbacks for grounded theory in that it is often falsely assumed to be easy to do. Robson and McCartan [2016, 163] pick up on similar drawbacks, where they claim grounded theory is "by no means an easy option, and not to be undertaken lightly". Additionally, Robson and McCartan [2016, 162] dismiss that no prerequisite theoretical ideas are necessary, arguing that it "is not possible to start a research study without some pre-existing theoretical ideas and assumptions". This supports the argument from Saunders et al. [2011] that adopting a grounded theory strategy is neither an excuse nor

reason to ignore appraising existing literature.

Suddaby [2006, 640] claims "the seamless craft of a well-executed grounded theory study, however, is the product of considerable experience, hard work, creativity and, occasionally, a healthy dose of good luck". These points are echoed by both Robson and McCartan [2016] and Saunders et al. [2011, 149] supporting the view that grounded theory is messy and far from perfect [Suddaby, 2006], necessitating researchers to "develop tacit knowledge of, or feel for, their data".

Ethnography

Ethnography owes its existence to anthropology, and is very much rooted within the inductive approach to research [Saunders et al., 2011]. Researchers adopting an ethnographic strategy are concerned with describing and interpreting culture and social structures observed within a group of individuals, doing so by immersing themselves as much as possible within that group and associated culture [Robson and McCartan, 2016].

Thus, ethnography is a research strategy that is inherently longitudinal given the time commitment required to undertake such studies [Saunders et al., 2011]. As with case studies, ethnographic research focuses on phenomena in context, but considerably differs given case study research still uses specific and prescribed techniques for data collection of which, ethnography asserts such methods are too simplistic to fully capture the complexities within social contexts [Saunders et al., 2011].

The term "naturalism" is applied to ethnographic studies, whereby the researcher is firstly, not only an active participant, but also conducts direct observation of other participants [Saunders et al., 2011, 150]. Saunders et. al. add that the naturalism can become confused given its meaning in Positivist research, where it is connected to the use of scientific methods and models in research.

Ethnography can be an effective strategy for descriptive and exploratory research purposes, and can be especially potent for deep description of the phenomena

as well as the culture and context within which it occurs [Geertz, 1973; Robson and McCartan, 2016]. However, ethnography is criticised with concerns that the researcher-participant relationship is too close, potentially giving rise to issues with the integrity, quality and validity of the research being undertaken [Robson and McCartan, 2016]. Logistics are another issue with ethnographic research in the social sciences as researchers first need to locate a setting or group that will allow sufficient access and over a potentially long period of time [Robson and McCartan, 2016; Saunders et al., 2011]. Furthermore, researchers need to build trust with each participant [Saunders et al., 2011] in addition to a solid understanding of the setting, including any informalities and jargon, let alone "specialist concepts used when talking about socio-cultural systems" [Robson and McCartan, 2016, 157].

While mitigative action can be taken to preserve research integrity, namely in the form of detailed, high-quality notes and records of researcher-participant interaction [Emerson et al., 2011], such action only adds to the inherent time consuming nature of ethnography [Robson and McCartan, 2016; Saunders et al., 2011]. Thus Saunders et al. [2011, 150] adds that researchers should, in addition, "develop strategies to cope with being both a full member of the social context in which the research is set, as well as undertaking the research".

Taken together, ethnography represents an incredibly flexible and potentially powerful research strategy for studying phenomena in context, but may be very difficult for new researchers given the risks, especially with regards to ethics and integrity, and as such should never be taken lightly [Robson and McCartan, 2016].

2.3.4 Research Strategy Selection and Justification

While these strategies can fall under a deductive or inductive approach, Saunders et al. [2011] argue that no research strategy should be considered inferior or superior to another. Aside from the systematic literature review, two distinct phases researching DevOps are defined and linked to the research questions. Furthermore, the research questions aid in determining the strategy [Robson and

McCartan, 2016], befitting of the pragmatic philosophical undertones for the research within this PhD thesis, with each phase adopting a different strategy as outlined in the following paragraphs.

A survey strategy was selected for answering the question of "How can DevOps be defined?" (RQ1). The survey strategy allows for the controlling of research activities, which was desired for answering RQ1. An experiment strategy also offers control, but to the degree where it would be too limiting in this research.

The case study strategy was selected for answering the remaining research questions (RQ2, RQ3 and RQ4). The literature highlights a distinct lack of research on the realities of DevOps in organisations. As a phenomenon, DevOps can be studied within the context it occurs and over time, therefore from a pragmatic perspective, the case study is a good methodological fit for answering these research questions. While data collection methods are outlined in this chapter, the case study allows for both mixed or multi-method techniques, thus allowing a researcher to analyse data by means of triangulation, where the researcher uses "multiple sources to enhance the rigour of the research" [Robson and McCartan, 2016, 171].

Ethnography would also offer a good methodological fit, especially with regard to understanding the social processes connected to DevOps adoption in an organisational context. However, logistically the commitment Ethnography demands was neither possible for the researcher nor organisation in this PhD research.

2.3.5 Technique Choices and Time Horizon

Three technique choices for data collection are available, mono-method, mixed-method and multi-method. Each technique can be applied to one of two time horizons, cross-sectional or longitudinal.

Mono-method, as its name implies is the technique of applying a single method for collecting and analysing data. It is especially common in experiment research strategies [Saunders et al., 2011].

The mixed-method technique utilises multiple methods, but where the type of data they acquire is different, namely quantitative and qualitative. Whereas the multi-method technique, while similar, acquires data of the same type [Saunders et al., 2011].

The time horizon refers to the overall picture the research shows. Saunders et al. [2011, 155] phrases the following question to eloquently explain the time horizons choice: "Do I want my research to be a snapshot taken at a particular time, or do I want it to be more akin to a diary or a series of snapshots to be a representation of events over a given period?" Thus, the time horizon in research falls into two categories: Cross-Sectional, where the research offers a snapshot of the topic being studied at a given time, or Longitudinal, where the research offers insight into the topic over a given period, thus time can become a variable of the research too.

For this PhD research, phase one, looking at the definition of DevOps is mixed methods and cross-sectional as it considers both qualitative and quantitative data collection and analysis methods. Whereas phase two, investigating the adoption of DevOps will be undertaken over a fourteen month period, but collecting and analysing qualitative data from both open format diaries and interviews, therefore making it multi-method and longitudinal.

2.3.6 Overview of the Empirical Work in this Thesis

The research presented within this thesis is exploratory in purpose with a pragmatism philosophical stance taken. This enables the research to be undertaken in two phases, designed in a manner which follows a deductive and inductive approach. Moreover, this research seeks to avoid dogmatic philosophical arguments, treating the approaches and strategies for their merits and applying them appropriately.

The first phase investigates the definition of DevOps and takes a predominately deductive approach due to the usage of definitions already present in the literature

(see section 3.3.1), which are utilised in a focus group of DevOps practitioners. The output of the focus group is then used within a questionnaire sent to participants within the DevOps community. This cross-sectional mixed methods approach produced both qualitative and quantitative data.

The second, and larger phase of the research studied the adoption of DevOps within a medium sized business. An inductive approach was taken here utilising the case study strategy, albeit from a Constructivist view, drawing on the works of Stake [1994] rather than the Positivist influences of Eisenhardt [1989] and Yin [2013]. Large amounts of qualitative data are collected through a diary study and a series of semi-structured interviews over a period of fourteen months.

The design of this research and how it maps from philosophy to time horizons can be seen in figure 2.2.

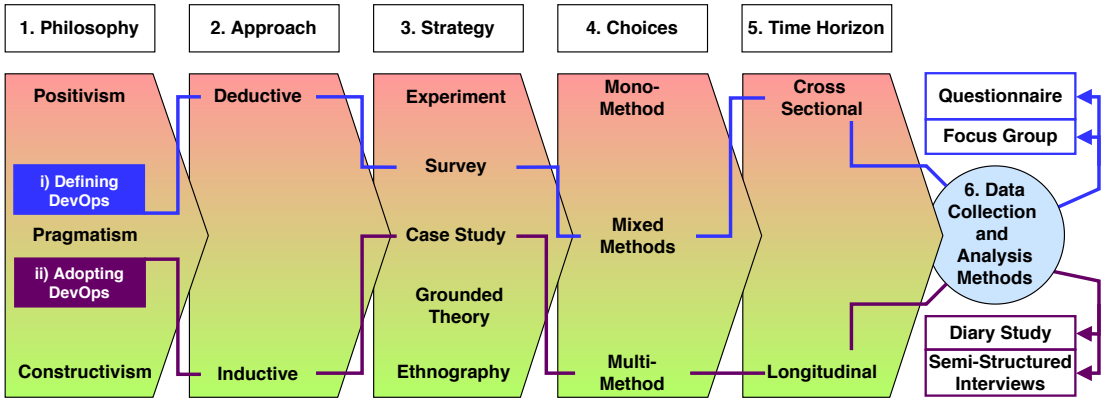


Figure 2.2: Research map for the constituent parts of the project.

The following sections outline and discuss the methods for data collection relevant to the element of the research they were utilised within. In addition, a systematic review of the DevOps literature is undertaken, forming a core component of the literature review, presented in chapter 3. The review method and protocol is presented in section 2.6.

2.4 Method for Exploring the Definition of DevOps

As introduced in section 1.1, a clear issue presenting itself in the literature is how DevOps is defined. While there is no shortage of definitions proposed for DevOps, it remains difficult to define given a lack of consistent definitions [Dyck et al., 2015].

For the purposes of this PhD, a deductive and mixed-method study consisting of a focus group and questionnaire was undertaken to explore the definition problem with industrial insight. Additionally, with the seeming difficulty in defining DevOps, this research aimed to establish a base line theoretical position in order to study DevOps in context.

2.4.1 Focus Group

A focus group, is a type of group interview, and as its name implies, specifically focuses on a particular issue [Saunders et al., 2011], in this case the definition of DevOps. A focus group relies upon interaction between the participants, enabling a consensus on the topic to be reached.

The objectives of the focus group in this research are threefold; firstly, to identify and agree on a set of core conceptual attributes which can inform any definition of DevOps. Secondly, to produce a new, or validate an existing definition for DevOps using the previously agreed attributes. Thirdly, to provide a means to assist in fine tuning the research activities to be undertaken when exploring the adoption of DevOps through a case study.

In line with the philosophy taken, the objectives will be met predominately through group activity, discussion and brainstorming. To meet these objectives, the focus group was structured into two exercises, with the participants split into two groups.

Participant recruitment was restricted to individuals who identified as having done, or currently work in a DevOps environment (perceived or otherwise) and have done so for two or more years. The group was made up of individuals from a range of roles within software engineering and IT departments.

A total of 12 practitioners were invited from a number of national and international organisations, seven of whom actually participated (see table 2.2). Additionally, the supervisory team assisted the PhD researcher in the facilitation of the focus group (see table 2.3). The PhD researcher oversaw the group, introducing each task and keeping timings, having no direct involvement with the participants when undertaking their tasks. The other two members of the supervisory team were each assigned to a group and were there to aid by means of scribing notes and ensuring the group maintained focus on the task. An agenda for the focus group is provided in appendix 1 on page 217.

Position/Role	**Organisation**	**Size**	**Sector**	**Group**
Systems Developer	University of East Anglia	Large	Education	1
Head of Research Computing	University of East Anglia	Large	Education	2
Chief Technology Officer	Tech Marionette	Micro	Tech	1
DevOps Engineer	Worldpay	Large	Finance	1
Senior Architect in Technology Operations	Worldpay	Large	Finance	2
Software Development Manager	Anglia Farmers	Medium	Agriculture	1
Senior DevOps Contractor	Unboxed Consulting	Small	Tech	2

Table 2.2: Participants of the DevOps focus group.

Name	Position/Role	Institution
Steve Jones	PhD Researcher	University of East Anglia
Fiona Lettice	Professor in Innovation Management	University of East Anglia
Joost Noppen	Principal Researcher	BT

Table 2.3: Hosting and facilitation team for the Focus Group.

2.4.1.1 Exercise One - Agree Core Conceptual Attributes of DevOps

The aim of this exercise is to identify and agree on a set of core conceptual attributes. This exercise is divided into three tasks, each with specific sub-objectives (see table 2.4). Two groups of participants (as set out in table 2.2) undertook the same tasks within this exercise.

#	Title	Description
1	Silent Brainstorm	Individual task to produce as many attributes of DevOps as possible. Timeboxed to 15 minutes.
2	Intra-Group feedback, discussion and prioritisation	Feedback and discussion within groups on the attributes produced previously. Assign a priority as "high", "medium" and "low" (H,M,L).
3	Inter-group feedback, discussion and prioritisation	Feedback and discussion across both groups on prioritised attributes. Both groups to agree a final, joint set of attributes.

Table 2.4: Tasks making up exercise one of the focus group.

For the silent brainstorm task, each participant was provided with post-it notes to write attributes on. The small size of the post-it notes encouraged concise answers and ease of moving into the following tasks.

For the intra-group feedback task, each group was provided a pre-prepared A1 sheet of paper with three columns drawn out: "H" (High), "M" (Medium) and "L" (Low). Additional blank A1 sheets were made available to each group if requested. Attributes produced in the previous tasks were discussed within each

group with an emphasis on firstly, agreeing the attribute and secondly, prioritising it by placing it within the relevant column on the prepared sheet (see appendix 2 on page 218).

In the final task of the first exercise, both groups came together with their list of prioritised attributes, to discuss, agree and prioritise an inter-group set of core conceptual attributes. A new, prepared A1 sheet of paper was provided for the final attributes to be placed. The final agreed list of prioritised attributes was given to the primary researcher to be included within the second exercise following a short break.

2.4.1.2 Exercise Two - Defining DevOps

With exercise one producing an agreed set of core conceptual attributes of DevOps, exercise two would focus on how DevOps is defined. The aim of this exercise is for each group to produce a definition for DevOps using the previously agreed attributes as a guide.

During a short break, the research team prepared the venue by placing nine definitions on the wall on one side of the room (see appendix 2 on page 218). These definitions are all taken from the literature (see table 3.1). Each definition was placed within a pre-prepared A1 sheet divided into two columns to represent the positives (+) and negatives (-) respectively.

Two approaches were designed to accomplish this exercise, with each group being assigned one each. Group one undertook the task of evaluating existing definitions in order to validate or derive a new definition based around the agreed attributes. Conversely, group two were asked to produce a new definition from scratch.

For the evaluation of existing definitions, group one were asked to score and/or discard any they uniformly disagreed with. This was accommodated by another section at the bottom of each A1 sheet.

The PhD researcher oversaw the exercise, maintained timing and did not interact

with the groups. The two members of the supervisory team continued to facilitate the same group as before, acting as scribes and taking notes.

The final output of the focus group were prioritised attributes and two definitions of DevOps. These would subsequently serve as input for further study through a questionnaire, as outlined in the next section.

2.4.1.3 Focus Group Limitations

Compared to other methods of data collection, a focus group has control implications over the data collected. This is due to the researcher having less control as compared to an interview, but also the open ended nature of such an activity. Furthermore, the sample of participants is small, thus potentially introducing problems in drawing generalisations from the data. Additionally, focus groups by their nature, are collective, and therefore a limitation is that any output is by virtue, collective [Robson and McCartan, 2016]. This is by no means a problem if the research seeks collective views, however, additional methods need to be considered if seeking individual outputs. For this research, a questionnaire was utilised following the focus group in order to consider individual views.

Focus groups need clear boundaries, and possibly moderators, with regards to the topic being discussed in order to prevent digression [Saunders et al., 2011]. As such, the focus group was structured into two specific exercises, both concerning the definition of DevOps. Additionally, the PhD researcher involved the supervisory team to act as moderators within the focus group, steering participant discussions and assisting with tasks such as note taking. This helped the primary researcher maintain impartiality during the exercises as well as continued focus on the core topic.

Another argument given by Saunders et al. [2011] is that of participant motivation. When involving business people in research activities, there often has to be some form of benefit for them. All of the participants desired to meet others involved with DevOps elsewhere, a situation which was leveraged by the PhD

researcher where following the focus group activities, a networking lunch was provided along with a presentation about DevOps from one of the participants.

2.4.2 Questionnaire Survey

The term questionnaire is generic, referring to all data collecting techniques where individuals provide responses to the same set of questions in a prescribed order [De Vaus, 2013]. While questionnaires are typically self-administered, i.e. where completion is undertaken independently, Saunders et al. [2011] and Gill and Johnson [2010] also assert that they can be administered by means of telephone or where an interviewer is present (see figure 2.3).

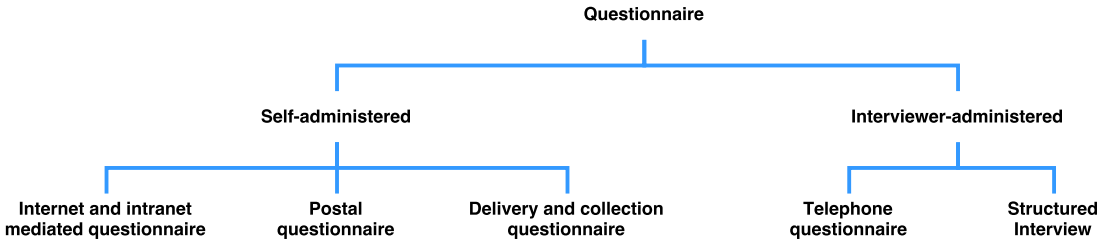


Figure 2.3: Types of questionnaire grouped by how they are administered [Saunders et al., 2011, 363].

Questionnaires are often used when researchers adopt the survey strategy and they work well for descriptive or explanatory research purposes especially for attitude and opinions [Saunders et al., 2011].

In this PhD research, a questionnaire is utilised to supplement the research activities undertaken within the focus group. This involved taking the output of the focus group and gauging the opinion of DevOps practitioners, but at an individual, rather than collective level [Robson and McCartan, 2016].

The focus group agreed on a total of 17 conceptual attributes of DevOps and produced two definitions. While this was a collective output, as discussed in the focus group limitations section, an internet mediated questionnaire was developed and sent to DevOps practitioners asking them to state their agreement with the

17 attributes and to specify a preference for one definition. In addition, they were asked to comment on each definition from both a positive and negative stance.

The overall and primary objective of the questionnaire was not to draw any final conclusions, but rather to explore agreement and/or disagreement with the attributes and definitions produced by the focus group. Moreover, the attributes would serve as set themes within the second element of the research project, exploring actual DevOps adoption in an organisation. The questionnaire was therefore a mechanism to fine tune an instrument for the analysis of a large quantity of qualitative data later in the project.

Questionnaire Structure and Distribution

The questionnaire was developed using Google Forms[1] and divided into four sections. The first section captured information regarding the respondent's opinions on DevOps, including 17 conceptual attributes identified and agreed by the focus group. Attribute questions within this section were also presented in a different order to each respondent.

The second and third sections asked each respondent to indicate a preference for one definition and what they perceived, both positively and negatively about each. Finally, the fourth section captured information about the participant including Job Title and if they are UK based or not. Of the 17 questions requiring completion, a total of 12 required a closed answer. Thus five questions within the questionnaire captured qualitative data around the positives and negatives of each definition and the job title of the respondent. A specimen questionnaire is provided in appendix 3 on page 220.

While Google Forms enabled the development of the questionnaire, it also allowed for easy electronic distribution. This was achieved through various local tech communities, including SyncNorwich[2] and Norfolk Developers[3]. In addition,

[1] Forms is part of the Google Drive suite (https://drive.google.com accessed: Jun 2017)
[2] https://www.meetup.com/syncnorwich accessed: Jun 2018
[3] https://www.norfolkdevelopers.com accessed: Jun 2018

social media was utilised, in particular, an item was posted within the LinkedIn pulse outlet, specifically asking DevOps practitioners to spare around 10 minutes to complete the questionnaire. As such, the researcher attempted to target individuals working within software development and IT systems support roles where DevOps is being practised and/or adopted as much as possible. The questionnaire was left live for a period of one month, and achieved 83 complete responses.

Limitations

It is acknowledged that questionnaires are much harder to produce and collect data with than they would appear [Gill and Johnson, 2010; Saunders et al., 2011] and difficult to entirely decouple the effects caused by length, topic and method of administration [De Vaus, 2013].

A recurring consideration for researchers undertaking questionnaires is that of sample size [Gill and Johnson, 2010]. Saunders et al. [2011, 581] adds that in experimental research, it is necessary to calculate a "precise minimum sample size". In this research, the population is unknown, rendering it impossible to calculate a minimum sample size. Secondly, and perhaps more crucially, the research strategy employed was not that of experimentation.

Additional mitigation to these limitations is the adoption of a mixed methods approach when dealing with this data. As has been previously stated, the questionnaire also contained questions of a qualitative nature, necessitating a different approach to the analysis. Thus, the questionnaire survey results did not wholly rely on quantitative approaches. Furthermore, the questionnaire instrument is a minor component of this overall study and was utilised to provide an initial set of themes for the later thematic analysis on qualitative data as part of a 14 month longitudinal study detailed further on in this chapter.

2.4.3 Data Analysis

While cross-sectional, both qualitative and quantitative data were produced by the focus group and questionnaire survey. In this section, techniques for analysing the data are presented.

Quantifying Agreement on DevOps Attributes and Definition

The attributes identified by the focus group were presented within questions 1.6, 1.7, 1.8 and 1.9 of the questionnaire. As agreement was sought from each respondent, a test for inter-rater agreement was undertaken using the Kappa coefficient. Given the reach of the questionnaire survey, the domicile of respondents (Question 4.2: UK or Non-UK based) is considered. To achieve this, Cohen's Weighted Kappa is utilised and outlined in the following paragraphs.

Cohen's Weighted Kappa considers agreement and disagreement across two raters [Cohen, 1968], as represented by the following formula:

$$\kappa_w = 1 - \frac{\sum_{i=1}^{k}\sum_{j=1}^{k} w_{ij} p o_{ij}}{\sum_{i=1}^{k}\sum_{j=1}^{k} w_{ij} p e_{ij}}$$

p_o = observed agreement. j = matrix column.

p_e = chance agreement. w = weighting.

i = matrix row.

The first thing to occur is the calculation of a matrix containing expected values or chance agreement figures which are obtained by multiplying the total rows by the total columns before dividing it by the total number of observations.

The weighted kappa coefficient takes into its calculation a predetermined matrix of weights (see table 2.5), which allow for calculation of disagreement, taking into account the chance and observed agreement. Weights are either linear, where

the difference between categories has the same importance; and quadratic where difference between categories varies in importance.

Linear Weights	1	0.75	0.50	0.25	0
Quadratic Weights	1	0.937	0.75	0.437	0

Table 2.5: Linear and quadratic weights for calculating weighted Cohen's Kappa across five categories.

In this analysis, linear weightings were used, given that there was no difference in importance between the categories. Kappa is then calculated as 1 minus the product of observed agreement before being summed with the product of the corresponding weights. Finally, the totals are divided by the product of chance agreement corresponding to the weights.

The value of Kappa is always less than or equal to 1, which according to Landis and Koch [1977], can be interpreted into six strengths of agreement as shown in table 2.6.

κ	**Strength of Agreement**
$<$ 0.00	Poor
0.00 – 0.20	Slight
0.21 – 0.40	Fair
0.41 – 0.60	Moderate
0.61 – 0.80	Substantial
0.81 – 1.00	Almost Perfect

Table 2.6: Kappa statistic strength of agreement [Landis and Koch, 1977, 165].

In determining an inter-rater agreement with regards to the size of the organisation a respondent works for, the weighted kappa coefficient cannot be used given its limitation to two raters. In this case, responses were grouped into four raters based on the size of the organisation by number of employees, as defined by the UK government definition of organisation size (see table 2.7.) The aim of this analysis was to gauge what level of agreement there was in relation to organisation size.

Size	No. of Employees
Micro	≤ 9
Small	10 - 49
Medium	50 - 249
Large	≥ 250

Table 2.7: UK Government definition of business size [Rhodes, 2016, 5]

For each attribute, the following propositions are considered:

P0 Agreement on the DevOps conceptual attribute is not different according to domicile.

P1 Agreement on the DevOps conceptual attribute differs according to domicile.

Elaboration Analysis

Elaboration analysis is a broad term for a number of methods utilised in the analysis of quantitative data typically gathered from within survey strategies, allowing the researcher to "explore the effects of other variables" [Robson and McCartan, 2016, 433]. One such methods of analysis undertaken for this PhD research is factor analysis.

Originating from Psychology, factor analysis branches from multivariate analysis, focusing on identifying any latent covariance and correlation between variables [Lawley and Maxwell, 1962]. As a method of analysis, factor analysis comes in two forms: confirmatory factor analysis (CFA) and exploratory factor analysis (EFA).

Robson and McCartan [2016] outline factor analysis as a tool to making sense of correlations between a number of variables. Thus Robson and McCartan [2016, 436] define factors as "hypothetical constructs developed to account for the intercorrelations between the variables". Subsequently, factor analysis offers the

researcher a means to turning a large and potentially unwieldy number of variables into a smaller number of easily manageable and understandable factors.

While CFA is a technique to verify factors, EFA, as it's name implies, is used to explore the data and identify potential factors arising from the correlation between variables [Robson and McCartan, 2016]. Therefore, EFA always begins with a generated correlation matrix between the variables. Robson and McCartan [2016] also state the number of variables should not exceed the number of respondents, and there should be five times the number of respondents to variables for reliably estimating these underlying factors.

An EFA will be undertaken with the agreements over 17 DevOps conceptual attributes (variables) considered by questionnaire respondents. The EFA began with producing a correlation matrix to aid in identifying strong correlations, which were further tested for significance. A CFA was undertaken on the identified factors for the purpose of verification by assessing how well the variables loaded. Finally, a Cronbach's Alpha test for reliability was conducted [DeVellis, 2016], as defined in the formula:

$$\alpha = (\frac{k}{k-1})(1 - \frac{\sum_{i=1}^{k} {\sigma^2}_{y_i}}{{\sigma^2}_x})$$

Where:

k = number of components. ${\sigma^2}_y$ = variance of component i

${\sigma^2}_z$ = variance of observed scores.

2.5 Method for Exploring the Adoption of DevOps

Little research has focused on the actual adoption of DevOps within an organisation, and what this means for both software engineering and IT support professionals. This component of the PhD research aims to study, in depth, the effects

and implications for both the business and software engineering functions as a result of DevOps adoption.

Lethbridge et al. [2005] argue that a multiple method approach to data collection is important when capturing information from software engineering professionals. Lethbridge et al. [2005] outline two techniques of data collection to consider when undertaking software engineering research (see table 2.8). This research adopts the approach advocated by Lethbridge et al. [2005] in order to capture data pertaining to revealing insight and understanding with regards to methods and processes, in addition to rich longitudinal and real-time insights for any phenomena under investigation.

However, careful considerations of method and design is needed to minimise or, if possible, avoid the so called "Hawthorne Effect" where participants deliberately change behaviour as result of being directly observed [Lethbridge et al., 2005, 317].

Technique	**Method Examples**	**Description**
Inquisitive	Focus groups; Interviews; Questionnaires; Conceptual modelling;	Good for providing general understanding and insights into methods and processes. Data collected generally offers point in time and cross-sectional insights.
Observational	Diary studies; Direct observation; Document analysis; Analysis of tool usage;	Can provide in-depth and real-time insights regarding any phenomena under investigation. Data collected generally offers rich and longitudinal insights.

Table 2.8: Inquisitive and Observational techniques for research involving software engineering professionals, adapted from Lethbridge et al. [2005, 313].

This section outlines a 14 month, multi-method study at Anglia Farmers Ltd. (AF), a medium sized organisation adopting DevOps. The study combines both the inquisitive and observational technique as highlighted by Lethbridge et al. [2005] with the objective of understanding DevOps within the context of the organisation, as well as any methods and processes in place. To accomplish this, a qualitative diary study is supported by a series of semi-structured interviews,

designed to probe deeper into insights that emerge from the diaries.

2.5.1 Open Format Diary Study

Diary studies are a method for the capturing of data at regular intervals, focusing on the actual participants and their behaviour within a situational context, whilst minimising the effects of actual observations [Carter and Mankoff, 2005] and therefore reducing the "Hawthorne Effect" [Lethbridge et al., 2005, 317].

Diary studies are most often structured methods designed to gather quantitative data [Ohly et al., 2010], but can however take an open format where the participant uses their own words, thus generating qualitative data [Poppleton et al., 2008].

An open format diary study covering a 14 month period, starting in January 2016 and ending in March 2017, was undertaken within the Software Development and IT Operations functions at AF. The diary study was qualitative and open reflection allowing for participants to report on events and experiences within the context and time frame of which they happen. This approach can therefore contribute to reducing retrospective bias [Reis and Gable, 2000], yet be ideal for exploratory research [Lethbridge et al., 2005].

Participants were asked to provide a diary every two weeks. In addition, a series of semi-structured interviews supplemented the diary study, as outlined in section 2.5.3.

Table 2.9 provides an anonymised overview of the participants within AF throughout the diary study.

2.5.2 Pilot Study and Abductive Reasoning of Job Crafting

A pilot diary study was undertaken by a senior software developer at AF in June and July 2015, with a total of five open format diaries written. Initial analysis of

Position/Role	Department	Diaries
Software Development Manager	Software Development	16
Senior Software Developer	Software Development	22
Senior Software Developer	Software Development	24
Software Developer	Software Development	5
Software Developer	Software Development	4
Software Developer	Software Development	5
Software Developer	Software Development	9
Test Analyst	Software Development	3
Test Analyst	Software Development	0
Business Analyst	Software Development	18
Systems Administrator	IT Operations	3
Systems Administrator	IT Operations	4
Head of Group Operations	Senior Management	0
	Total Diaries:	**113**

Table 2.9: Diary study participant overview at AF.

the data showed not only the effectiveness of using Bitbucket[1] as a repository for submitted diaries, but revealed unexpected links with the theory of job crafting (see Wrzesniewski and Dutton [2001]). In particular, large amounts of task and cognitive job crafting were evident in the diaries, which were further confirmed in a debriefing of the pilot study with the participant.

This abductive finding quickly became a focus of the study, providing a solid anchoring with business management theory and thus adding weight to the argument that DevOps is not exclusively a Computer Science topic. Section 3.4 gives a brief introduction and overview of the theory of job crafting. Additionally, potential barriers with regards to using new tools in the adoption of DevOps were evident.

The data collected during the pilot study was also considered within the analysis

[1] https://bitbucket.org/ accessed: Jun 2018

of diary and interview data collected during the main study.

Manner of Data Collection

The researcher wanted to consider the individual participants within their environments when applying methods and techniques as asking individuals to commit to a 14 month study is no small task. Such consideration is especially pertinent with software engineering professionals [Lethbridge et al., 2005]. As such, the diary study was designed around the tools used on a daily basis by software developers at AF, in this case, Bitbucket, which is software platform developed by Atlassian Software[1], and is widely used for the version management of source code in software development projects.

Bitbucket accomplishes this through the git[2] protocol. Git works through the staging and committing of updated content which is then pushed to the repository. As such diaries were "committed and pushed" to the repository for easy researcher access (see figure 2.4) but also to the rest of the team, thus in keeping with the Constructivist view of knowledge generated through social processes. The added benefit is that the diary study harnesses existing processes and skills typically utilised day to day by the participants.

The diary templates participants were provided were written using Markdown[3], a lightweight markup language which is easy to read and write. Markdown files are easily identified by the extension '.md'. Markdown is a markup language, and offers the benefit of often being rendered within a web browser in conjunction with a platform such Bitbucket. For software development professionals, Markdown is often the format for which documentation such as installation guides and readme files are written. The template provided to the participants, along with guiding questions used is provided in appendix 4 on page 224.

While this mechanism proved useful, and indeed engaged software developers,

[1] https://www.atlassian.com/ accessed: May 2017
[2] https://git-scm.com/ accessed: May 2017
[3] https://daringfireball.net/projects/markdown/ accessed: Jun 2017

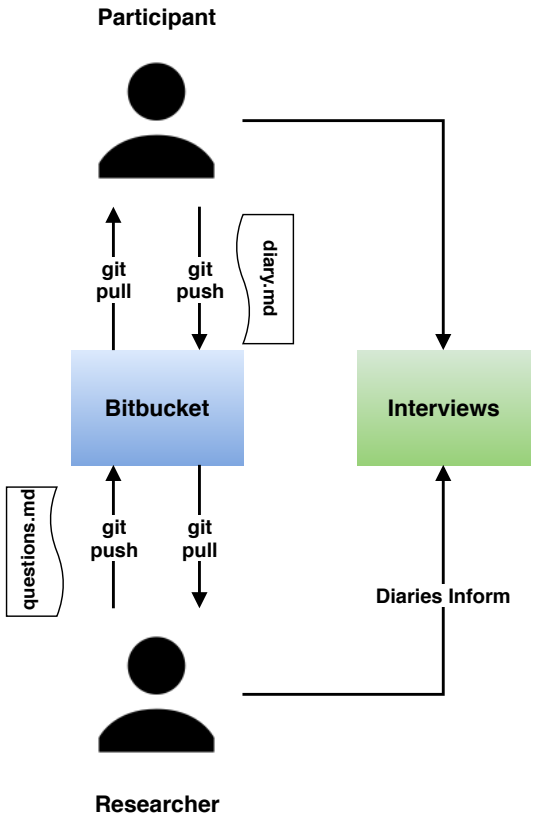


Figure 2.4: Process of using Bitbucket for submitting open reflection markdown diaries with participants at AF.

participants within the IT Operations function at AF neither use Bitbucket, nor have the necessary expertise to use it. As such, diary collection from them was a more manual undertaking, generally via e-mail submissions to the researcher. In this case, the diaries were committed to the Bitbucket repository by the researcher, with the prior permission of the participants.

While the majority of diaries were open and viewable by all participants through the Bitbucket repository, some diaries were submitted privately and directly in order to retain anonymity. This was down to participant request and often where more sensitive things were discussed within them. These diaries were never committed to the repository but still considered in the analysis process, as outlined in section 2.5.4.

Limitations

Gaining insight into individuals, their roles and what this means on a day to day basis in software engineering environments, diaries make for a potentially excellent choice in method [Lethbridge et al., 2005]. However, there are limitations needing consideration, especially if the time horizon is longitudinal.

The first consideration is participant attrition, where participants may submit diaries infrequently or stop altogether. In this study, some attrition was observed, namely from the AF software development team. Upon investigation, the participants concerned generally felt what they would be contributing 'would not be of interest' subsequently believing they were wasting the researcher's time. However, they did not formally withdraw from the study and willingly participated in the interviews. As such, the researcher attempted several remedies for them, including transcribing verbal diaries, by means of recording reflection according to the guiding questions. Thus, these so called verbal diaries had more in common with an interview and were not completed independently by the participant.

While these verbal diaries provided some remedy, they added logistical complexity to the project. A total of four participants resorted to the verbal approach, meaning 17 diaries in total were submitted in this fashion. The researcher did however run the transcribed diaries past each participant, with them providing agreement before they were added to the repository (or kept private as per participant desire). While not ideal, this method did enable data to be captured and kept said participants engaged with the study.

2.5.3 Semi-Structured Interviews

Saunders et al. [2011, 318] cite Kahn and Cannell [1957] in defining interviews as the "purposeful discussion between two or more people". Interviews can be undertaken in a formal or informal manner [Gill and Johnson, 2010].

In practical application, they can be categorised into three types, namely struc-

tured, semi-structured and unstructured [Robson and McCartan, 2016; Saunders et al., 2011]. Interviews can be a rich source of data given the potential to probe and follow up on answers, especially in face-to-face situations [De Vaus, 2013]. Table 2.10 provides an overview of each interview category, along with the research purpose for which it is suited.

Type	Format	Suitability	Data
Structured interviews	Fixed Questionnaire	Descriptive Explanatory	Quantitative
Semi-structured interviews	Flexible questions themes	Exploratory Explanatory	Qualitative
Unstructured interviews	non-directive	Exploratory	

Table 2.10: Types of interview with links to research purpose derived from Saunders et al. [2011].

Structured interviews are standardised, and generally involve the interviewer directly administering a questionnaire comprising a predetermined set of questions [Saunders et al., 2011]. There is little to no room for flexibility, so interviewers would read questions precisely as they are written and ideally in the same tone of voice with all participants. Responses tend to be predetermined options, therefore structured interviews would typically produce quantitative data.

Semi structured interviews may follow some elements of the structured interview, but do not conform to a standard [Saunders et al., 2011]. Aside from a list of questions, semi-structured interviews may also include themes to explore, enabling flexibility for the interviewer to adjust the order, add or even omit questions as necessary [Robson and McCartan, 2016; Saunders et al., 2011]. As a result, each semi-structured interview is unique given the ability for the interviewer to make adjustments as the discussion evolves. They produce qualitative data, usually recorded by an audio recording and/or through note taking [Saunders et al., 2011].

By definition the opposite of structured, unstructured interviews are informal and non-standardised with no predefined questions or themes [Saunders et al., 2011].

They allow for in-depth discussion but the interviewer would still need to steer this discussion in relation to what they seek to explore. As with semi-structured, unstructured interviews are unique to each participant and produce qualitative data which, as with semi-structured interviews, would be recorded and/or written [Saunders et al., 2011].

With the three categories of interviews introduced, the actual interview itself can take many more forms including, but not limited to: interviewer-administered questionnaires; face-to-face; telephone; and group-based interviews (see figure 2.5). Indeed, focus groups, as explored in the previous sections of this thesis are a type of group-based interview.

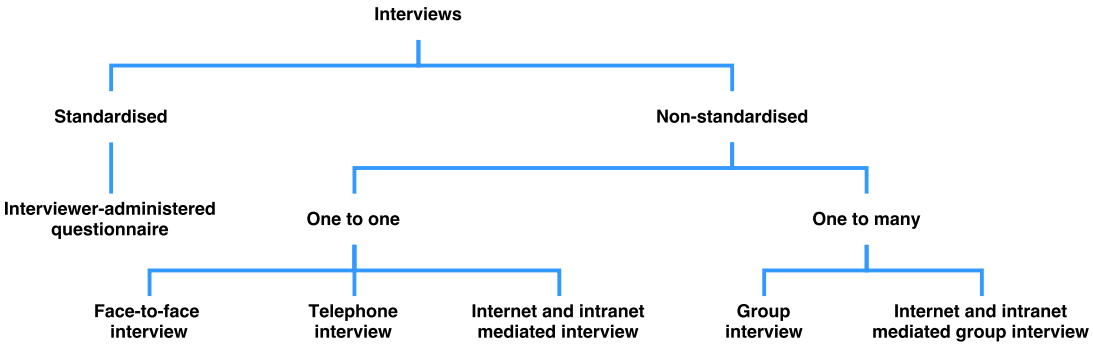


Figure 2.5: Forms of interview grouped by how they are conducted, adapted from Saunders et al. [2011, 321].

Semi-Structured Interviews to Supplement Diary Study

To supplement and support the diary study, and applying Lethbridge et al.'s (2005) recommendations of a multi-method technique when dealing with software engineering professionals, four semi-structured interviews are undertaken with each participant during the study. The purpose of these interviews is to capture more detailed information, especially with regards to methods and processes around DevOps adoption. Additionally, the interviews further explore the participant as an individual within these processes, enabling the researcher to further probe insights emerging within the diaries submitted by the participant

being interviewed.

While the semi-structured approach makes each interview unique to the participant, some degree of structure was established. The interview structure was piloted and refined with two software developers at Rainbird[1], a small technology company based in London and Norwich, UK.

The following paragraphs will now outline the location of the interviews for participants, along with the levels of preparation undertaken. Unlike the diary study, interview recordings and transcriptions were stored privately and away from Bitbucket.

Conducting the Interviews

AF kindly allowed the use of meeting rooms at their offices in Honingham Thorpe, Norfolk. This helps minimise disruption to the organisation, yet offers convenience for participants, who also had ready access to amenities. Each participant was made aware of the interview around two weeks in advance and was aware that any previously submitted diaries were examined and potentially probed.

Prior to the interview commencing, the researcher will engage in small talk with the participant, getting to know them better and discussing shared interests, especially with subjects related to technology in general. The aim of this is to put the participant at ease, generate rapport and to project an informal atmosphere for interview itself.

Participants are interviewed up to four times over the fourteen month period (see figure 2.6). Given the time commitment involved, alternate interview scenarios have been planned and considered, as illustrated by the three alternate paths. Such scenarios included staff starting with or leaving the organisation and those where circumstances were prohibitive. For instance, it is only possible to interview a Senior Manager at AF twice in the period of study.

[1] http://rainbird.ai/ accessed: Jun 2017

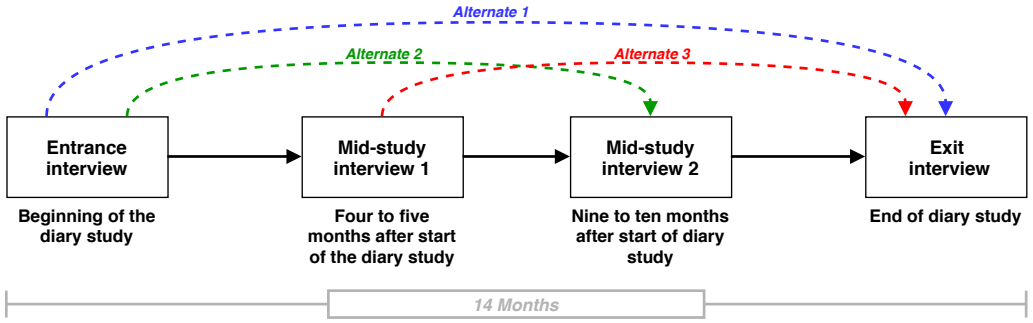


Figure 2.6: Multi-Interview plan for participants at AF over a fourteen month period.

All participants, at the very least, have an entrance and exit interview, and these were considered the most important of all. By the end of the study, a total of 44 semi-structured interviews will have been conducted with participants from AF (see table 2.11).

Position/Role	**Department**	**Interviews**
Software Development Manager	Software Development	4
Senior Software Developer	Software Development	4
Senior Software Developer	Software Development	3
Software Developer	Software Development	4
Software Developer	Software Development	4
Software Developer	Software Development	4
Software Developer	Software Development	3
Test Analyst	Software Development	3
Test Analyst	Software Development	2
Business Analyst	Software Development	3
Systems Administrator	IT Operations	4
Systems Administrator	IT Operations	4
Head of Group Operations	Senior Management	2
	Total Interviews:	**44**

Table 2.11: Interview participation at AF.

The purpose of the entrance interviews is to discuss their initial perceptions about DevOps, how they see their role at AF, including how it compares to previous ones and what they do on a daily basis. While more structured than others, the entrance interviews are informal and the discussion sufficiently flexible, allowing the researcher to probe, add, change or remove questions as they deemed fit. The entrance interviews typically take 30 - 40 minutes to complete. A generic protocol for the entrance interviews is provided in appendix 5 on page 227.

The mid-study interviews have fewer initial questions, instead, focusing on probing entries of interest from submitted diaries. They are to also enable the researcher to check in with each participant individually, and investigate diary absences and address any concerns they have with the research. These are typically shorter than the entrance interviews, taking between 25 and 35 minutes to complete. Given the study length, participants have two mid-study interviews planned. A generic protocol for the mid-study interviews is provided in appendix 6 on page 230.

The exit interview concludes the diary study and affords the opportunity to debrief each participant individually. Aside from some specific questions, the exit interviews continue to probe participant submitted diaries. As such, these interviews are slightly longer, taking between 40 and 60 minutes to complete. A generic protocol for the exit interviews is provided in appendix 7 on page 232.

Limitations

Interviews can provide a good insight, but not necessarily a full and accountable observation of what happens in software development environments [Lethbridge et al., 2005], and thus limit the longitudinal time horizon taken with the study as a whole. The interviews outlined in this section are designed to supplement and support a diary study.

Taken together, the interviews and diary study constitute a multi-method approach, which Lethbridge et al. [2005] argue is necessary to acquire the fuller

picture in such environments. Furthermore, the interviews act as a control mechanism for the diary study, which in turn fuels the discussion points.

Saunders et al. [2011] warns that with non-standardised interviews, interviewer bias is an ever present danger, which can be mitigated by a good recording of the interview. Fortunately, all participants were happy with the interview being recorded, and this was accomplished with an Olympus DM670 voice recorder, with each recording transcribed in an intelligent verbatim manner, namely, omitting 'erms' and 'ahs', which were deemed to not add anything; long pauses, were however transcribed as was laughter. Additionally, the interview transcriptions aided in a thematic analysis, in conjunction with the diaries as is outlined in section 2.5.4.

2.5.4 Data Analysis

Both Saunders et al. [2011] and Robson and McCartan [2016] highlight that the analysis of qualitative data, even in relatively small amounts can very easily overwhelm researchers. Robson and McCartan recommend four possible artefacts (see table 2.12) which can potentially aid researchers in keeping track of qualitative data, and the key things within it.

All four of Robson and McCartan's suggested artefacts are applied, with some modifications. By the nature in which the Mid-Study and Exit Interviews probed participant diaries, they necessitate the ongoing analysis and interim reporting of these diaries, therefore providing an implicit benefit through the inherent application of the document sheets and interim summary. Furthermore, the adoption of these artefacts, while laborious, will enable ongoing analysis of large quantities of qualitative data.

A thematic analysis is an approach to analysing qualitative data, which can be applied in a Constructivist manner whereby events, realities and meanings can be derived from discourse captured within a sociocultural context [Robson and McCartan, 2016]. Coding is key to undertaking thematic analyses, and Robson

Activity	Description
Session Summary	Summarising the key points on what has been obtained. As the name implies, this can be a very useful activity for sessional research activities, such as interviews, focus groups and observations.
Document Sheets	Like with session summaries, but applied instead to each document allowing the technique to be applied to non-sessional activities such as diaries.
Memoing	An overarching term that applies to capturing anything throughout the research project. This is a useful technique for abductive reasoning and capturing ideas, views and any other intuition through all stages of analysis.
Interim Summary	As the name implies, this is an attempted summary of findings at a specific point in time. The interim summary allows the researcher to consolidate what has been found, to potentially highlight what needs to be found and how these relate to the research questions.

Table 2.12: Rundown of recommended artefacts which can aid in the analysis of qualitative data according to Robson and McCartan [2016, 467].

and McCartan [2016, 467] define these as "passages of text or other data items such as the parts of pictures that, in some sense, exemplify the same theoretical or descriptive idea". These codes are subsequently grouped into a smaller number of 'themes', also referred to as "categorisation of meanings" [Saunders et al., 2011, 490], relating to the research questions. These themes or categories can be predefined ahead of the analysis (set themes) or they can emerge (emergent themes) during [Robson and McCartan, 2016].

For this PhD research, a thematic analysis of the qualitative data collected from the interviews and diary study was applied, using both set and emergent themes. The set themes (see table 2.13) were derived from the output of the focus group and questionnaire exploring the definition of DevOps, with an emphasis placed on management themes. Additionally, the three types of job crafting as put forward by Wrzesniewski and Dutton [2001] were set themes.

To fully account for themes, and in keeping with the exploratory purpose of this

Theme	Description
Decision Making	Any and all aspects of management decision making within a software engineering environment.
Ownership	The concept of taking ownership of the development of specific features within a software system either individually or collectively.
Responsibility	The concept of taking responsibility for the development of specific features within a software system either individually or collectively. Particular emphasis is placed on the notion of shared responsibility from development to deployment of software.
Measurability/Metrics	The focus on metrics measuring the success of the DevOps approach being employed by the organisation.
Accountability	The provision and/or recognition of accountability for actions taken within a software engineering environment.
Task Crafting	Any instances where participants change the task boundaries of their roles and/or role meanings change as a result of tasks.
Relationship Crafting	Any instances where participants rethink their relationships with colleagues, seeing their role as a part of an integrated whole.
Cognitive Crafting	Any instance where participants examine and rethink their role as more than just delivering outputs.

Table 2.13: Set themes for use within a thematic analysis of qualitative diary and interview data collected at AF

research, emergent themes were explored in addition to the set themes. While software packages such as NVivo can potentially help in such analyses, the thematic analysis was undertaken manually. Additionally, interrelated themes were identified, especially linking job crafting with the management and software engineering themes. Interpretation of the findings is presented within a case study, providing a narrative of DevOps within the context of AF as well as highlighting the temporal aspects of the study.

Case Study Formation and Boundary

Adopting the Constructivist view of Stake [1994, 38] where the case study is the "study of the particular", a qualitative case study was written concerning DevOps adoption within AF. The case study boundary can therefore be defined, but in no ways "fixed" as the Software Development and IT Operations functions, within which DevOps is being adopted as part of AF's new approach to software development.

2.6 DevOps Systematic Review

A literature review's purpose is the coherent presentation of key findings from relevant primary studies, to a wider academic and practitioner audience [Tranfield et al., 2003]. However, the ubiquity and availability of literature can create problems for researchers [Jesson et al., 2011], especially when dealing with a "mass of often contradictory evidence" [Tranfield et al., 2003, 207].

2.6.1 Introduction to Systematic Literature Reviews

Having roots in healthcare and medical research [Biolchini et al., 2005; Greenhalgh et al., 2005; Tranfield et al., 2003], the Systematic Literature Review (SLR), often shortened to 'Systematic Review' can be defined as "a comprehensive review of literature which differs from a traditional literature review in that it is conducted in a methodical (or systematic) manner, according to a pre-specified protocol to minimise bias, with the aim of synthesising the retrieved information" [Dempster, 2011, 15].

In short, the SLR is a meticulous and methodologically explicit secondary study activity [Clarke, 2011; Kitchenham, 2004], differing considerably from the traditional (inductive and deductive) approaches employed for reviewing literature [Hanley and Cutts, 2013; Jesson et al., 2011; Tranfield et al., 2003]. The ad-

vantage for researchers is that the SLR can be a powerful research instrument given its inherent rigour, structure and reproducibility owing to its methodological transparency [Hanley and Cutts, 2013; Kitchenham, 2004; Tranfield et al., 2003].

Published SLRs are often observed having multiple authors [Jesson et al., 2011], suggesting a SLR would be a considerable (but not necessarily impossible) task for an individual. Undertaking a SLR (see figure 2.7) is typically a linear activity [Jesson et al., 2011]. The first stage is to scope the area of study, defining the review questions and producing the protocol which will drive the SLR. Following this, literature is searched, screened and documented according to the protocol, before more thorough analyses of each item's quality is undertaken. Data is then extracted from included items and synthesised, resulting in a coherent and critically written review document [Hanley and Cutts, 2013].

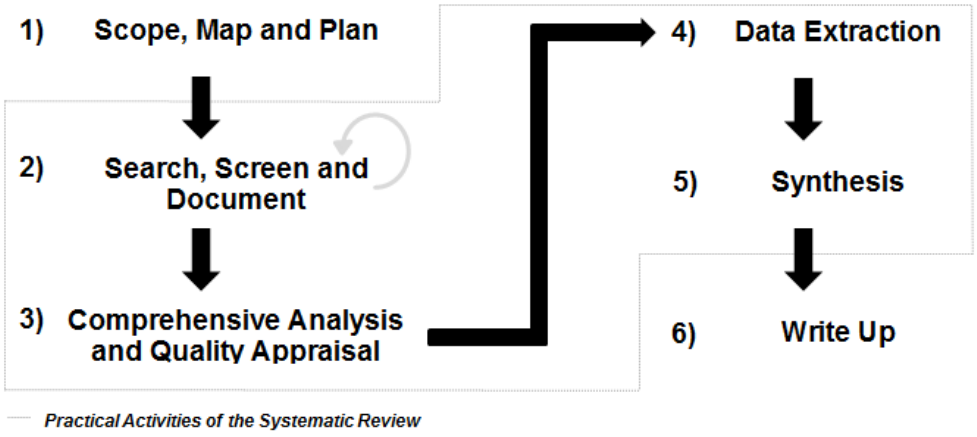


Figure 2.7: Overview of the key stages of a SLR, derived from Hanley and Cutts [2013, 4], Clarke [2011, 64] and Jesson et al. [2011, 103-104]

Application in Management and Software Engineering Research

Despite its roots in Medicine [Jesson et al., 2011], the SLR has been successfully utilised in both the Management and Software Engineering disciplines [Biolchini et al., 2005; Kitchenham et al., 2009; Tranfield et al., 2003].

While the process of undertaking a SLR is fundamentally the same (see figure 2.7), the strict application and rigid adherence to a pre-defined protocol, may render SLRs counter-productive in management [Tranfield et al., 2003] and software engineering [Kitchenham et al., 2009]. As such, the protocol-driven search and inclusion activities are criticised for compartmentalising the review, leading to bias, the very thing SLRs aim to prevent [Greenhalgh et al., 2005]. However, Tranfield et al. [2003] argue the protocol-driven search is a key attribute of the SLR, and that disciplines such as Business Management, which are often exploratory, can overcome these limitations by not excluding researcher intuition, knowledge, networking and serendipity. Indeed these aspects can be easily included within a SLR, supplementary to the protocol.

2.6.2 Protocol

Following initial scoping revealing a general lack of research activity on DevOps prior to 2010, a SLR is undertaken to appraise the growing DevOps literature. The SLR process follows a protocol, which considers the application of a SLR outside of the Medicine discipline. As such, the final protocol draws heavily from the work of Biolchini et al. [2005] and Greenhalgh et al. [2005]. It also incorporates the recommendations by Tranfield et al. [2003] and Kitchenham [2004], allowing for researcher knowledge, networking, snowballing and serendipity in locating relevant literature. Thus further accommodating the exploratory nature of Business Management and Software Engineering research (see figure 2.8).

The final deliverable of the SLR is twofold. Firstly, this section was developed to serve as a methodological overview, providing transparency and reproducibility. Secondly, a detailed and comprehensive review of the found literature is conducted, written up and presented in section 3.3.

The purpose of the review is to provide a collated overview of the DevOps literature and to explore its application within the Business Management and Software Engineering disciplines. The review aims to advance the discussion on DevOps, given the need for further research, for what is an emerging topic, as concluded

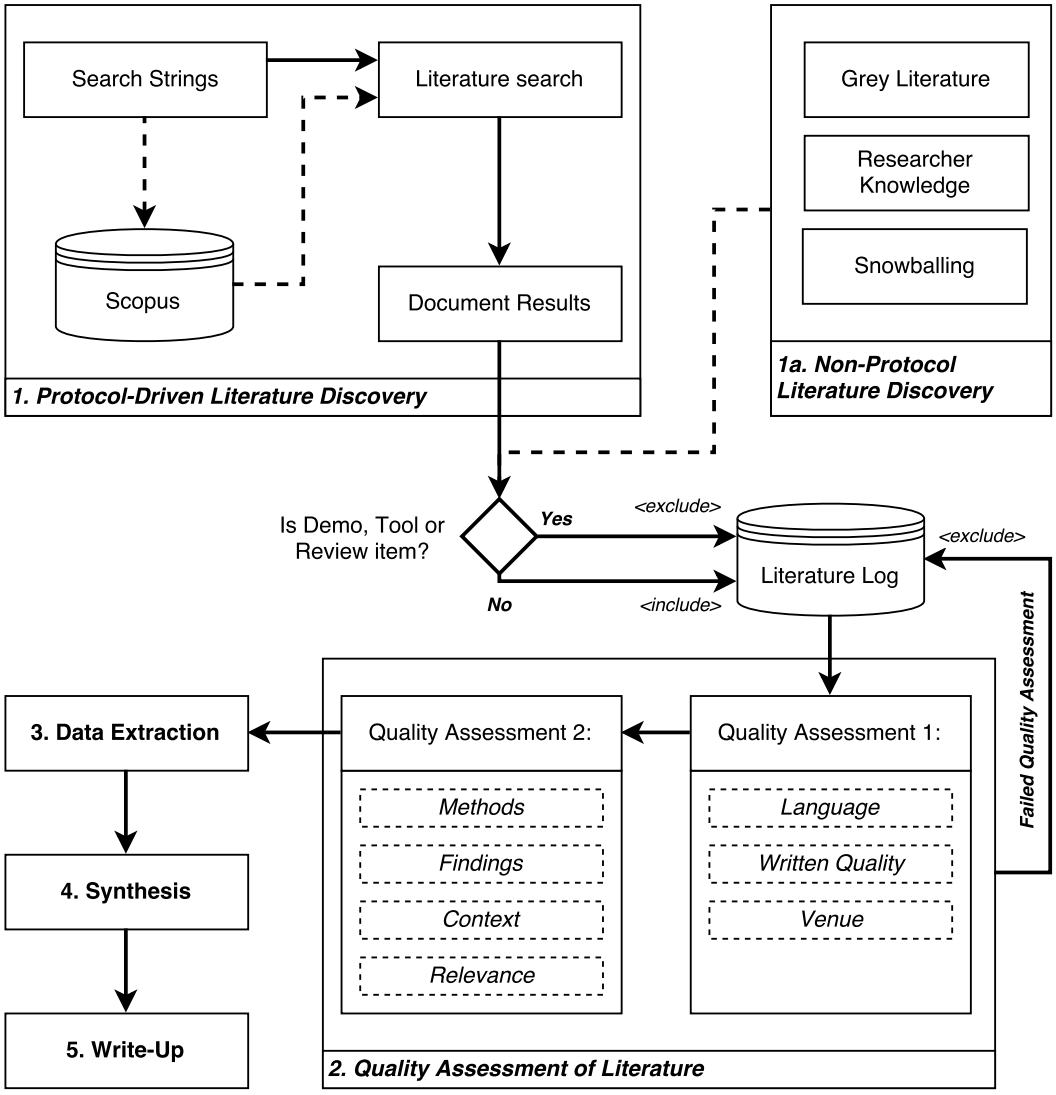


Figure 2.8: Process for the SLR on DevOps

by Erich et al. [2014].

Initial Scoping Searches for DevOps

In the first instance, the term "DevOps" was searched in 2015 with the initial results from Scopus producing 220 peer-reviewed items. The first of these items was published in 2010, but has increased to 1842 publications by the end of 2019.

Google Scholar, unsurprisingly, produced the most items in the initial scoping

search, followed by Scopus. While Google Scholar is an attractive option, it was not selected as a primary source for literature in this review. Instead, Scopus was selected given it offers the same search functionality and customisation options that Google does. Furthermore, Scopus has an interface with UEA's library and so also produces a list of items it holds (or can access via interlibrary lending).

While Scholar indexes many of the main publishers, as well as grey literature (see section 2.6.2 for more details on grey literature inclusion), Scopus has a far stronger focus on peer-reviewed literature. Nevertheless, Google Scholar is useful for locating further literature for the vast majority of items found.

The SLR seeks answers to the following questions:

1. How can DevOps be defined?
2. How does DevOps affect the organisation?
3. What is the current research agenda for DevOps?

Searching, Screening and Documentation of DevOps Literature

Initial keyword and terms were identified, forming the basis of the search strings used for the review. These terms were chosen given the previously defined review questions and the output of the definition research activities (see section 2.4). In total, 14 strings (see table 2.14) were used to search the title, abstract and keywords of items on Scopus[1] given that it indexes many publishers, including: ACM; Elsevier; Emerald; IEEE; Springer and Wiley. The term "DevOps" is integral, and was included in every search string.

Although all strings produced results, much of the results in later searches are duplicates already discovered previously. In addition, some of the search strings utilised phrase searching, as denoted by the use of double quote (" ") characters. This enabled a more precise search, especially with terms such as 'deci-

[1] https://www.scopus.com/ accessed Jun 2020

Search String	
DevOps	DevOps AND Agile
DevOps AND Definition	DevOps AND "Software Development"
DevOps AND "Software Engineering"	DevOps AND Infrastructure
DevOps AND Business Management	DevOps AND Strategy
DevOps AND Culture	DevOps AND "Decision Making"
DevOps AND Metrics	DevOps AND Ownership
DevOps AND Accountability	DevOps AND Responsibility

Table 2.14: Strings used to search for peer-reviewed DevOps literature using Scopus.

sion making', 'software development' and 'software engineering'. Despite being largely dominated by computing and software engineering research, published peer-reviewed research on DevOps drastically increased from 2015 (see table 2.15 and figure 2.9).

2010	2011	2012	2013	2014	2015	2016	2017	2018	2019
1	11	14	20	49	125	232	326	485	579

Table 2.15: Peer-reviewed DevOps literature found on Scopus by year.

Each item of literature found is recorded on a spreadsheet, allowing for the capture of additional data about the item as it progresses through the process. Additionally, a bibliography is kept with each item given a unique reference for ease of future citation. Again, Google scholar was instrumental in quickly providing citations in the correct syntax, although accuracy was ensured by applying manual corrections to the bibliography as needed.

Quality Appraisal

In total 87 peer-reviewed items were shortlisted for quality assessment, with 35 selected for inclusion in the final review. The majority of rejected items were tool demonstration papers and initial analysis shows that DevOps is emerging in the literature; more evidently so within Computer Science. However, very little appears to be written about DevOps within the Business Management literature.

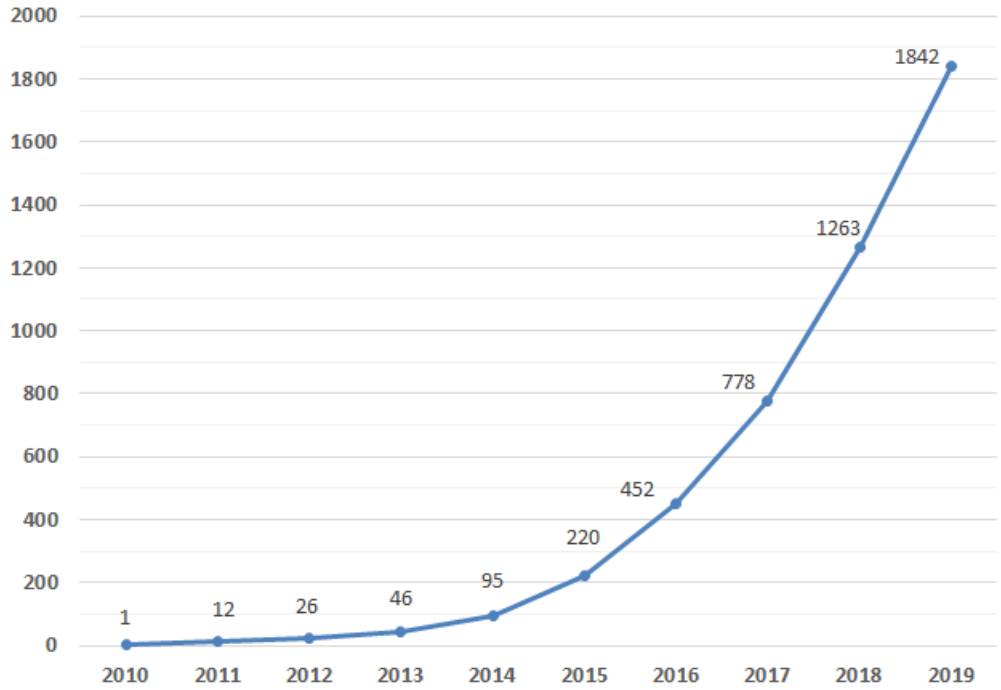


Figure 2.9: Cumulative frequency of peer-reviewed DevOps literature published from 2010 to 2019 inclusive.

Irrespective of how individual literature items are discovered, each is still subject to the same quality appraisal, which in this case, is broken down into two distinct phases. The first phase assesses the language, quality of writing and clarity of the item, with the second considering the research methods, ontology and epistemology.

Only items written in English are considered, and the log was updated to track how much of the item was read in order to understand the general message it was conveying. The publication outlet was evaluated independently of its respective discipline. This is done using the Chartered Association of Business Schools (CABS)[1] ratings or impact factors. This thesis seeks to present DevOps as an interdisciplinary topic covering Business Management, Computer Science and Software Engineering. By omitting any discipline, the review questions would be out of scope, and as such would potentially compromise the review's integrity.

[1] https://charteredabs.org/ accessed: May 2017

However, while data on the outlet's rating is collected, the review itself did not discriminate purely on this rating. Indeed, disciplines consider ratings very differently with the CABS rating taking precedence within Business Management whereas the impact factor is prime consideration for Computer Science and Software Engineering. Sticking to one discipline's rating scheme may fail to consider relevant research published in outlets rated high in the other discipline.

Furthermore it is argued that a Journal or Conference rating, can and often does, implicitly influence a researcher given their perceptions of quality may be determined around this rating alongside other factors including author notoriety [Tranfield et al., 2003]. As such, and irrespective of research philosophy, such influence should not affect the appraisal of relevant, peer-reviewed literature.

Following on from the previous quality assessment stage, and continuing to follow recommendations from Tranfield et al. [2003], the research methods should be meticulously scrutinised in conjunction with the stated research question(s). As such, only primary studies are considered for the review. Where items are literature reviews, they are excluded at this stage. An exception is made if the literature review was a component of a primary study, as such the review section of the item is considered, but only for a contextual assessment of the primary study and associated research methods, where applicable. While excluded, peer-reviewed literature reviews are still counted in a meta analysis of found literature. This is to provide an indication on how well reviewed the DevOps literature is

Grey Literature Inclusion

As illustrated in previous sections, DevOps is an emerging field with a growing body of peer-reviewed literature. Industry on the other hand could be described as the tip of the spear with DevOps, with a huge and diverse amount of industrial literature available. As such, this already sizeable body of grey literature should not be dismissed, firstly because much of it is grounded in the sense that DevOps is predominantly being led by industrial adoption. Secondly, Adams et al. [2015, 187] argue that the "average lag" of about four years from publication to prac-

titioner application is another key justification for including grey literature. It follows that the exclusion of such content may jeopardise the completeness of the literature search.

A search for *grey literature in systematic reviews* reveals numerous articles across all disciplines that have successfully incorporated grey literature. However, a clear and explicit strategy for handling grey literature is required. Such a strategy has been adapted from Adams et al. [2015] as outlined in table 2.16, sitting well with the earlier considerations from Tranfield et al. [2003].

Strategy	Description
Authority and Reputation	The authority of the publishing institution, for example, for example, chartered institutes, reputable industrial sources, leading blogs
Expert Recommendation	Includes suggestions by prominent DevOps authors, speakers and practitioners.
Snowballing	Typically backward snowballing citations from both peer-reviewed and non-peer-reviewed items.

Table 2.16: Grey Literature strategy, adapted from Adams et al. [2015].

Data Extraction, Synthesis and Write Up

Literature items that clear the quality appraisal are considered "included" within the review. The next stage is to extract the key data from the literature item, which would include the research questions, further method details, key findings and any calls for further work, where applicable. Extracted data were analysed and synthesised, identifying key themes and concepts which would progress through to the methodology, research questions and design.

The synthesis stage enables the identification of research gaps and interdisciplinary themes across business management, computer science and software engineering. Additionally, this enables the precise refinement of the objectives and subsequent focus of research activities, selection of appropriate methods and par-

ticipant recruitment, pertinent to the project. Finally, the review was written up for inclusion in this thesis, as a systematic DevOps component of the overall literature review chapter.

2.6.3 Limitations

While a SLR can offer a powerful research instrument for appraising relevant literature, it is acknowledged that the undertaking of the SLR in this PhD research has limitations with one researcher leading to potential subjectivity and bias with reviewing literature. A mitigation to this limitations is to undertake such reviews with other researchers, thus helping to minimise any bias or subjective appraisals on quality.

2.7 Summary of Methodology

In this chapter, the research methodology of this PhD project has been presented. The PhD research is exploratory in purpose, taking a pragmatist philosophical position with both deductive and inductive approaches.

DevOps is a nebulous and difficult to define term, and as such, a cross sectional, mixed method study was defined under the survey research strategy with the aim of exploring the definition of DevOps. The study involved a focus group of DevOps practitioners as well as a questionnaire survey designed to test the output of the focus group. The resulting quantitative data were analysed by means of inter-rater agreement and elaboration analysis, involving exploratory and confirmatory factor analysis. These results are presented in chapter 4 on page 99.

Additionally, a longitudinal, multi-method study following a case study strategy was undertaken to explore DevOps adoption in a medium sized UK business. This study was conducted over a 14 month period, and utilised an open format

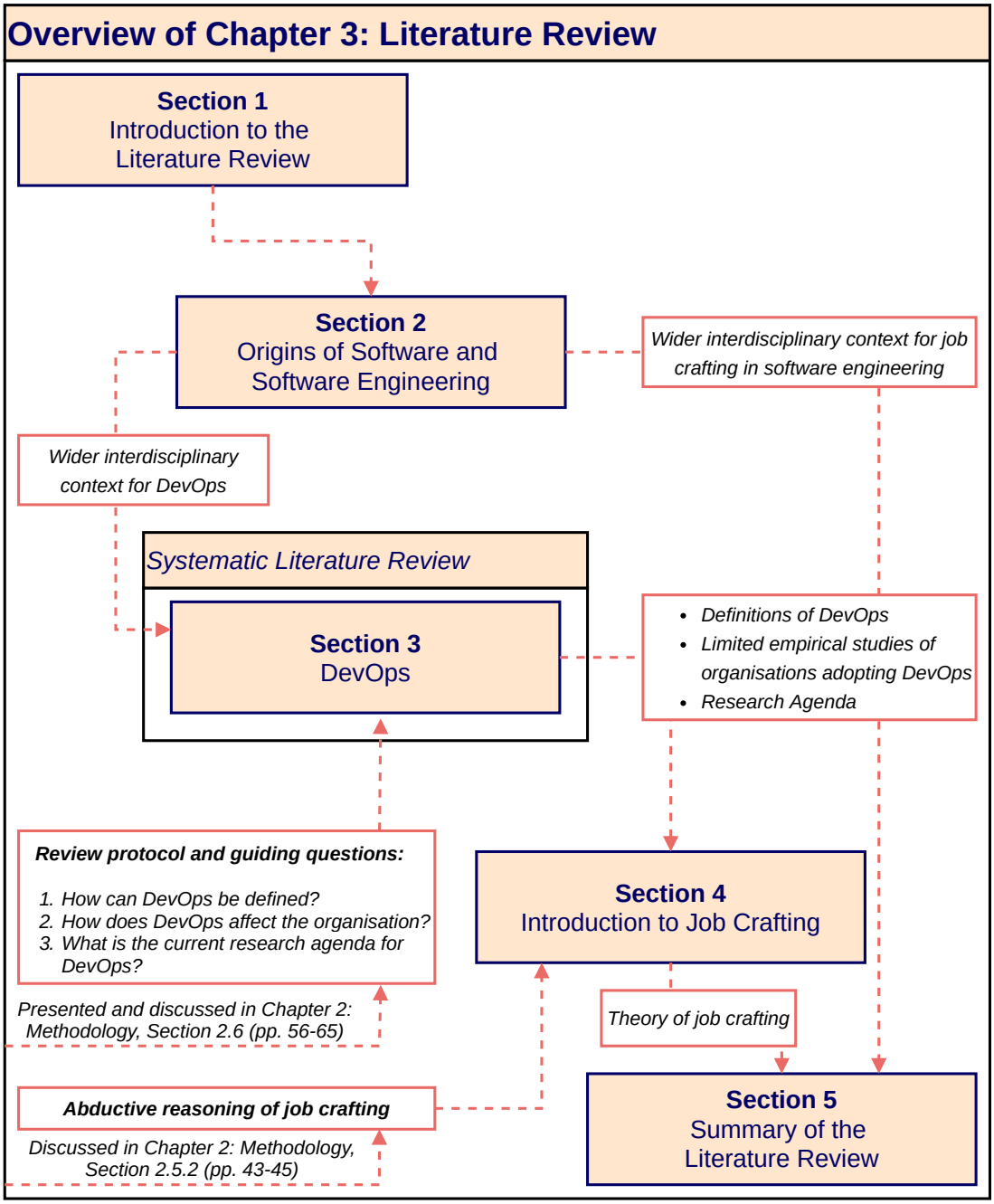
diary study with a series of interviews, specifically designed to probe insights from submitted diaries. As a result, a large quantity of qualitative data were collected and analysed using a thematic approach. Abductive reasoning following a pilot study revealed job crafting as a theoretical lens, and as such, the three types of job crafting as proposed by Wrzesniewski and Dutton [2001] were included as themes for analysis. The case study is available in chapter 5 on page 115. As part of the overall Literature Review, a SLR of DevOps literature was outlined and discussed along with a protocol to drive it.

Chapter 3

Literature Review

"The more I study, the more insatiable do I feel my genius for it to be."

– Ada Lovelace


Overview of Chapter 3: Literature Review
Section 1
Introduction to the Literature Review
Section 2
Origins of Software and Software Engineering
Wider interdisciplinary context for job crafting in software engineering
Wider interdisciplinary context for DevOps
Systematic Literature Review
Section 3
DevOps
Definitions of DevOps
Limited empirical studies of organisations adopting DevOps
Research Agenda
Review protocol and guiding questions:
1. How can DevOps be defined?
2. How does DevOps affect the organisation?
3. What is the current research agenda for DevOps?
Section 4
Introduction to Job Crafting
Presented and discussed in Chapter 2: Methodology, Section 2.6 (pp. 56-65)
Theory of job crafting
Abductive reasoning of job crafting
Discussed in Chapter 2: Methodology, Section 2.5.2 (pp. 43-45)
Section 5
Summary of the Literature Review

3.1 Introduction to the Literature Review

This chapter comprises three main sections, starting with section 3.2, which provides a brief overview of the origins and evolution of software and its development from the late 18th, 19th and 20th centuries. This also considers the business management implications of software development, and in turn, setting the context for DevOps.

Section 3.3 focuses exclusively on DevOps, providing a systematic literature review (SLR) which seeks to examine the growing body of DevOps literature. The aim of this core component of the chapter is to bridge the disciplines of Business Management, Computer Science and Software Engineering as well as explore what DevOps is, what it means for organisations and what, if any, interdisciplinary research agenda exists. The method and protocol for the SLR is available in the Methodology chapter of this thesis (see section 2.6).

Section 3.4 introduces and provides an overview of the theory of job crafting and its potential application to DevOps research, following abductive reasoning during the pilot diary study (see section 2.5.2).

Finally, a summary brings together the key insights from the literature indicating how they assist with informing the design of this PhD research.

3.2 Origins of Software and Software Engineering

Software can be defined as a set of programmed instructions which are executed by a computer for specific tasks, and is often simply referred to as a 'program'. Software varies in both scope and complexity, from a simple one line script to output a line of text to complex safety critical systems. Today, software is ubiquitous and applied in many areas including business, healthcare and transport.

The origins of software can be traced back to the late 18th and 19th centuries with the first examples of programming, including the works of Joseph Marie Jacquard

(1752-1834) and Augusta Ada King (1815-1852), better known as Ada Lovelace [Randell, 1994]. Inventor, mathematician and mentor to Lovelace, Charles Babbage (1791-1871) was inspired by the work of Jacquard when he designed the Analytical Engine in 1837 [Fuegi and Francis, 2003], making it one of the first designs for a general purpose computer. Lovelace recognised that aside from numbers, the Analytical Engine could manipulate symbols [Lovelace, 1843]. Subsequently, she wrote what is considered to be the first computer program which was a recursive algorithm to compute Bernoulli Numbers, referred to as 'Note G' (see appendix 8 on 234). 'Note G' was never implemented due to the Analytical Engine never being built to full-scale [Fuegi and Francis, 2003]. Nevertheless, symbolic logic remains a foundational concept of computer programming meaning Lovelace is recognised as one of the earliest programmers [Hollings et al., 2018].

Almost a century later, the work of Alan Turing (1912-1954) at Bletchley Park was instrumental in breaking the German Enigma during the Second World War. It was at this time that Turing shaped his own theories with regards to Computer Science and Software [Turing, 1937, 2009]. Turing was inspired by Lovelace's work, and laid the foundations for Computer Science as an academic and practical discipline [Hally, 2005].

In the decades following the end of the Second World War, Computer Science has both accomplished, and had much demanded of it. Development of the transistor enabled computer systems to evolve at a staggering rate, unlocking potential beyond the limits of previous valve and vacuum tube based systems [Friedman and Cornford, 1989]. While the transistor ushered in a new era for Computer Science, computers remained prohibitively expensive for most.

Nevertheless, businesses began to utilise the power offered by computers, initially for relatively simple data processing but increasingly for more powerful and complex applications [Barrow, 1999]. This not only necessitated businesses to create their own software, but also to adapt their management approaches as technology application in business continued to evolve in both scale and scope [Friedman and Cornford, 1989].

3.2.1 The Software Crisis and Software Engineering

In the 1960s, Intel co-founder Gordon Moore [1965] made a prediction that the number of transistors within integrated circuits will double every two years, and will continue to do so for the foreseeable future. This prediction, colloquially known as 'Moore's Law', means that the power of computers will increase rapidly.

As the decade continued, computational power increased by several magnitudes, which subsequently enabled software to increase in scale and scope [Dijkstra, 1972]. However, software development at this time was typically undertaken by skilled individual programmers, often referred to as hackers.

These individuals usually worked alone, lacking any structure to their work with little to no documentation rendering it difficult for others to maintain any previously developed software [Ince, 1988]. A lot of dependence was placed on this early software and the costs for redevelopment were often financially unfeasible, yet the challenges presented from its use and maintenance remain to this day [de Vasconcelos et al., 2017]. Indeed, Anquetil et al. [2007] report that 40 to 60 percent of maintenance effort is spent on understanding software.

With software increasing in complexity yet lacking any formal approach, the maintenance of developed software became a major problem to be tackled. This was subsequently labelled the "Software Crisis" or "Software Gap" during the First NATO Software Engineering Conference in 1968 [Randell, 1996, 70].

In reflecting on engineering and science at NASA during the Apollo program Rayl [2008] states that Margaret Hamilton, who is well known for her work on the Apollo on-board flight software, "developed the building blocks for modern "software engineering", a term Hamilton coined". While Computer Science is concerned with the study of design, theory and experimentation with computers, Software Engineering on the other hand is multi-disciplinary, yet purely concerned with software and the systematic application of engineering principles to developing it [Sommerville, 1992].

Although much discussion focused around the actual performance of software and

user expectations, Randell [1996] argue the consequences of software failure are increasing in severity for organisations. Larger, more complex or safety critical software systems where loss of life could be one such consequence, exacerbates the magnitude of software failure.

Furthermore, the Software Crisis was linked to increasing complexity in processes for developing software and advances in computer hardware. Problems manifested with software not being delivered because of projects running over time and budget; developed software was inefficient and of low quality; requirements and specifications were not met and left unfulfilled. Boehm [1988, 61] briefly summarises the earliest approach to software development as a "code-and-fix" model. This describes the act of writing code and only fixing problems if and when they occurred. No prior requirements analysis was undertaken. Boehm [1988] also claims that much of the software developed was a structural mess becoming increasingly expensive to maintain or was outright rejected by users.

The Traditional Approach

A proposed solution to the Software Crisis was a "metamorphosis in the practice of software production" [Randell, 1996, 73]. Hamilton and Zeldin [1976, 9] also argue that "formalized methodology" was crucial to software reliability.

In the late 1960s and early 1970s, the first structured frameworks for software development emerged, with Waterfall becoming the most known of these [Bell and Thayer, 1976; Royce, 1970]. Under Waterfall, software development is prescribed a series of linear stages (see figure 3.1), beginning with analysis and ending with maintenance. It asserts that software development activity can only move forward once all of the current stage activities are complete.

Waterfall always begins with analysis, where the requirements of the proposed software are gathered from stakeholders before being examined in detail. The requirements are formalised in a specification which may also be prioritised, forming a basis for designing the software. Again, detailed analyses are undertaken on

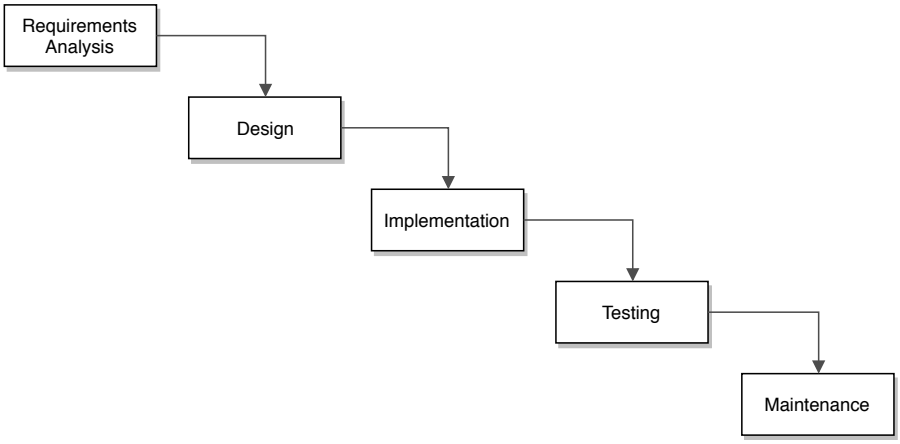


Figure 3.1: The Waterfall model, adapted from [Sommerville, 1992, 5].

the requirements and specification, leading to a comprehensive design which can then be implemented. Following implementation, testing occurs which will seek to verify the software works according to the specification and to validate it against stakeholder expectations. Finally, the software is operated and maintained where additional issues that arise are dealt with through support or addressing technical issues.

Waterfall became a widely used framework for software development as it appeared to tackle many of the difficulties encountered previously, especially with effectively capturing requirements and using these to guide software development [Boehm, 1988; Smidts et al., 1998].

Agile

By the late 20th century, Waterfall was acknowledged to have critical flaws, with many projects overrunning and producing unsatisfactory deliverables [Rubin, 2012]. The Standish Group[1] are well known for publishing their 'Chaos Reports', which report high levels of failure with traditional software development approaches. There is wide debate amongst researchers on the validity of these reports with critics focusing their arguments on the lack of methodological

[1] https://www.standishgroup.com/ accessed: Dec 2019

transparency [Eveleens and Verhoef, 2010; Glass, 2006; Jørgensen and Moløkken-Østvold, 2006]. However, this does not detract the rigidity of Waterfall, rendering it assumptive of human behaviour. Therefore it is often unable to cater for complex and rapidly changing requirements present in the majority of software development projects [Bell and Thayer, 1976; Boehm, 1988]. As a result, Waterfall can subsequently introduce technical failures in software itself [Smidts et al., 1998].

Brooks [1987] acknowledges Moore's Law as a suitable predictor for hardware advances, but he argues it does not apply to the development of software, given it does not advance in such the same manner. Brooks [1987, 11] put this down to the speed in technological advances where "the anomaly is not that software progress is so slow, but that computer hardware progress is so fast". However, the complexity and scale of software does increase in following hardware developments [Bosch and Bosch-Sijtsema, 2010]. Critically, Brooks [1987, 11] argues there is no silver bullet, neither technical nor managerial, that can eliminate the issues caused by increased complexity of software; a theme which Hamilton [2018] also mentions.

Brooks [1987] does however, advocate an iterative and incremental approach to developing software, with the 1990s seeing the emergence of various iterative and incremental methods, including but not limited to, Dynamic Systems Development Method (DSDM), Scrum and Extreme Programming (XP). As the name implies, these methods focus around a concept of frequent and short iterations of activity, thus breaking down a software development project into a number of smaller parts and therefore differing considerably to Waterfall. Figure 3.2 provides an overview of the Scrum framework, showing how work is broken down into backlogs and how time-boxed iteration is core to all activity, resulting in a viable deliverable.

At the beginning of the 21st century, seventeen software developers met to discuss these emerging iterative and incremental methods. The Manifesto for Agile software development was published as a result [Beck et al., 2001], formalising Agile as an approach to software development. Agile promotes team empowerment and

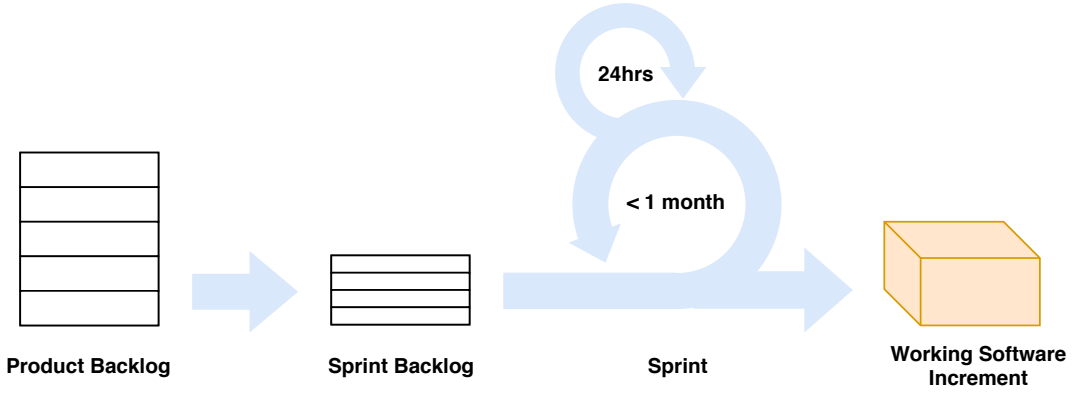


Figure 3.2: The Scrum framework, derived from Rubin [2012, 17].

self-organisation with a strong focus on collaboration and communication while following four core values [Šmite et al., 2010]:

Individuals and Interactions over processes and tools.
Working Software over comprehensive documentation.
Customer Collaboration over contract negotiation.
Responding to Change over following a plan.

It is important to note that while greater emphasis is placed on the bold points above, some believe Agile dismisses everything else [Brown, 2013]. Beck et al. [2001] do not however dismiss processes, tools, documentation, contracts and planning; but rather argue that while often a necessity, they should not take precedence.

Agile was a welcome means to overcome the limitations of traditional approaches to software engineering. However, the organisational adoption of Agile proved difficult for many years, particularly in large organisations and software development teams [Qumer and Henderson-Sellers, 2008]. Agile does however work well for smaller, co-located teams of typically less than ten developers [Boehm and Turner, 2003]. Bosch and Bosch-Sijtsema [2010] also found that smaller software development projects have benefited most from implementing Agile approaches.

Discussion is ongoing in the literature concerning adoption of Agile practice,

but several developments have arisen. In particular, how Agile works across distributed teams, its adoption for large-scale software development [Bosch and Bosch-Sijtsema, 2010] and its influence on business success [Martini et al., 2013].

3.2.2 Summary of Section

In comparison to other disciplines, Software Engineering is relatively young. Nevertheless, and alongside Computer Science, it has made significant contributions, as evidenced with the ubiquity of software today.

Early software development generally lacked any form of structure or method. The Software Crisis was precipitated by the wider consequences of the increasing complexity of software and a general lack of any formal approach to developing it. In response, early software development approaches were recognised and adopted, including Waterfall followed by iterative and incremental approaches such as Scrum. Eventually formalised under the heading of Agile, iterative and incremental methods have gained greater popularity, but traditional approaches such as Waterfall continue to be used as well.

Over time, effort has been made to ensure that software could be developed more systematically while minimising risk. Although Waterfall and Agile approaches were key developments, a new approach has recently emerged, called DevOps, which is explored further in section 3.3.

3.3 DevOps

This section builds upon the previous overview of how software engineering has evolved. As section 3.2 concluded, DevOps has emerged as a recent approach to software development. This section provides a more detailed overview of DevOps through a systematic review (SLR) of DevOps literature, published within the last decade following a protocol and guiding questions set out in section 2.6. A

total of 35 publications were included in the review (see appendix 9 on page 235).

The review firstly focuses on how DevOps is currently defined before looking into what DevOps means for organisations as well as the discipline and practice of Software Engineering. Finally, the current agenda for DevOps research is explored and highlighted before a summary is provided.

3.3.1 What is DevOps?

DevOps appears difficult to define, with many unclear, ambiguous and sometimes contradictory definitions [Dyck et al., 2015; Smeds et al., 2015]. According to Roche [2013], perspectives of DevOps are based upon one of two themes. Firstly, DevOps as a role with respective job descriptions and titles, for example, DevOps Engineer (see appendix 13 on page 255) and, secondly, DevOps as an emerging concept that addresses the needs and demands of modern software development. Furthermore, Roche [2013] argues that these themes are polarised, with one generally disagreeing with the other. Ghezzi [2017] argues that DevOps practice is not mature, often informal and unstructured; while Fokaefs et al. [2017, 25:2] claim DevOps "eliminates the concept of a software life-cycle as a system undergoes changes with no interruptions to consumers".

Another view is that DevOps differs from Agile given its focus also includes quality and operations alongside development, whereas Agile focuses on development [Gupta et al., 2017]. Yet, a contradictory view from Fokaefs et al. [2017] claims DevOps is solely focused on software development. Veres et al. [2019, 106] conclude that DevOps is a "concept that eliminates the barriers between traditionally isolated groups of developers and experts who operate the system, integrating them into a single complex team."

With a myriad of differing and often contradictory perspectives of DevOps, it is a term seemingly difficult to define [Smeds et al., 2015]. Nine definitions were discovered in the literature (see table 3.1). Most of these definitions share themes of team harmonisation, automation and the rapid deployment of software, but

each definition has a slightly different emphasis and form. Despite the frequency of research output increasing, DevOps remains under-represented in the literature and in need of a clearer definition [Airaj, 2017; Fitzgerald and Stol, 2017; Gupta et al., 2017].

#	**Definition and Source**
1	"The 'DevOps' approach to system administration introduces best practises from software engineering". [Obstfeld et al., 2014, 577]
2	"A set of practices intended to reduce the time between committing a change to a system and the change being placed into normal production, while ensuring high quality". [Bass et al., 2015, 4]
3	"DevOps is a movement within software engineering that professes to bring software developers and operations staff (those in charge of infrastructure, quality control, packaging, and release of software products) in close alignment, to ensure harmonious tasking and smooth transition of project artefacts through interoperable processes and tools". [Cois et al., 2014, 2]
4	"The DevOps movement addresses the gap between developers and operational teams in enterprise networks by borrowing techniques from agile programming practices, building tools that automate well-known manual steps". [Császár et al., 2013, 456]
5	"A cultural movement combined with a number of software related practices that enable rapid development". [Walls, 2013, 1]
6	"A set of engineering process capabilities supported by certain cultural and technological enablers". [Smeds et al., 2015, 170]
7	"A practice aimed at repairing the schism between the two teams." [Hosono, 2012, 330]
8	"DevOps is an evolution in thinking with regards how IT services are delivered and supported. It is a continuation of some of the predecessor work in the areas of continuous integration and application life cycle management (ALM); therefore, it is rooted in the agile philosophy, which also attempts to bridge the traditional organizational process divide between development and operations teams". [Mohamed, 2015, 51]
9	"DevOps is an organization approach that stresses empathy and cross-functional collaboration within and between teams - especially development and IT operations - in software development organizations, in order to operate resilient systems and accelerate delivery of changes". [Dyck et al., 2015, 3]

Table 3.1: Definitions of DevOps present in literature.

3.3.2 Organisational DevOps Adoption

Frequently changing requirements, contexts and market conditions are typical drivers for the rapid development and release of software. Despite Agile approaches enabling software developers to better respond to change, completed software is often passed to an IT Operations function, which deals with its release and support [Lapham, 2014; Mohamed, 2015]. Being able to rapidly release good quality software is a major motivator for many organisations [Bass et al., 2015]. It is this "unexploited potential of an IT division to increase value for the overall organisation" that Pass and Ronen [2014, 80] refer to as the "software value gap". Koilada [2019] claims that organisations adopting DevOps are early adopters. Furthermore, Koilada [2019] argues DevOps facilitates business model restructuring due to architecture innovation which it enables.

3.3.2.1 Continuing Evolution of Software and Development Approaches

Software also continues to evolve, with the emergence of microservice architectures which seeks to decompose software applications into individual constituent services which make up a whole. Thus microservices, as their name implies, are typically small software components, easing overall maintenance and complexity [Ranchal et al., 2015]. Microservices are becoming particularly common in cloud-based applications such as those offered by AirBnB and Netflix [Oliveira et al., 2016].

While the granularity and maintainability of microservices can be attractive, it can be very difficult to decompose a legacy software system into a series of microservices. Legacy software is still predominant in many organisations [Chen, 2017]. Moreover, Chen [2017, 82] argues that these systems are "usually not amenable" to Continuous Deployment and therefore it is very difficult to adopt a DevOps approach in dealing with them [Lwakatare et al., 2019]. Put simply, Chen [2017] refers to the process of software development itself and how a culture can be determined by the methods and technology used.

Therefore, organisational culture plays a part in this problem too as organisations often not only find it difficult to move on from such systems, but also from the processes and cultures of software development and maintenance which have formed around them [Airaj, 2017; Chen, 2017; McLarnon et al., 2014; Roche, 2013; Sebastian et al., 2017].

While DevOps can positively transform software development productivity and efficiency, it does this through organisational change [Pass and Ronen, 2014]. Ghezzi [2017, 9] argue that change "is often viewed as an afterthought rather than as a foundation principle" when considering the development of software. This is already evident with the numerous studies and reviews of traditional approaches to development available in the literature. However, change has to go beyond simply implementing new requirements within the software itself.

Managers also need to consider the impact such changes may have on the organisation, its operations and processes, especially where software is of strategic importance. Subsequently, management practice itself needs to evolve and adapt to changing conditions in order to accommodate and respond to change arising from software development activities [Sebastian et al., 2017].

DevOps seeks to mitigate issues introduced by such change through the functional harmonisation of both software development and IT operations.

3.3.2.2 DevOps and Digital Transformation

Continuing development and evolution of technology represents both an opportunity and threat for organisations. From the mid 1990s to the present day, software has changed substantially arising from infrastructure technological developments [Roche, 2013]. However, driven by Agile, and its ability to scale, software development methods and practices have continued to evolve too [Kneuper, 2017; Lapham, 2014; Roche, 2013].

While the definitions of DevOps vary, Smeds et al. [2015] have identified a number of capabilities and enablers with DevOps (see table 3.2). These are broken down

to three distinct categories.

Capabilities refers to technical processes which includes continuous integration and release of software. However, Smeds et al. [2015] argue that these are carried out continuously, needing constant feedback to inform them. But alone, these are not sufficient for DevOps adoption, but rather need the support of a compatible organisational culture with shared goals, ways of working and good communication to support increased automation of any process [Lwakatare et al., 2019]. Therefore a number of cultural and technological enablers are presented, which Smeds et al. [2015] argue should work in harmony with the capabilities.

Capabilities	Continuous Planning Collaborative and continuous development Continuous integration and testing Continuous release and deployment Continuous infrastructure monitoring and optimization Continuous user behaviour monitoring and feedback Service failure recovery without delay
Cultural Enablers	Shared goals, definition of success, incentives Shared ways of working, responsibility, collective ownership Shared values, respect and trust Constant effortless communication Continuous experimentation and learning
Technological Enablers	Build automation Test automation Deployment automation Monitoring automation Recovery automation Infrastructure automation Configuration management for code and infrastructure

Table 3.2: DevOps capabilities and enablers [Smeds et al., 2015, 171].

In a study investigating how large organisations tackle the challenges of digital transformation, Sebastian et al. [2017, 199] identified two distinct strategies, namely, "customer engagement" and "digitized solutions", which can help guide such organisations through technology-driven change or "digital transformation". Business strategy has evolved to leverage the opportunity presented by technology advancements, and Sebastian et al. [2017, 201] argue that an Operational

Backbone and Digitial Services Platform are "two technology-enabled assets" (see table 3.3) critical in the successful execution of strategy.

	Operational Backbone	**Digital Services Platform**
Management Objective	Business efficiency and technology reliability	Business agility and rapid innovation
Architecture Principles	Standardized end-to-end business processes; transparency into systems; data access	Plug-and-play business and technology components
Data	Single source of truth for transactional data	Massive repositories of sensor/social media/purchased data
Key Processes	Roadmaps; architecture reviews	Cross-functional development; user-centred design
Delivery Method	Fast Waterfall / regular software releases/SaaS adoption	Agile and DevOps; use of MVP (minimum viable product) concepts and constance enhancements
Funding	Major project / program investments	Continuous funding by business owners

Table 3.3: Management practices for operational backbone and digital services platform assets, taken from Sebastian et al. [2017, 205].

Sebastian et al. [2017] state "the operational backbone supports efficiency and operational excellence" and "the digital services platform supports business agility and rapid innovation". Of particular interest to this thesis is the identification of "Agile and DevOps" in the delivery method portion of the Digital Services Platform asset. This has arisen from observing two organisations, Permanente and Amazon Web Services (AWS), who have adopted a DevOps approach for the continuous delivery of software, resulting in substantially reduced innovation cycle times.

Further support comes from Chen [2017] and Sun et al. [2016], who both argue that Continuous Delivery (CD) and Continuous Integration (CI) bring huge benefits to organisations. CD and CI refer to the concepts of developing software in

short cycles of usually one month or less, testing it and ensuring it can be reliably released at any time. It follows that both CD and CI are intrinsic to DevOps practise [Gupta et al., 2017; Karl et al., 2016; Sun et al., 2016], and Sebastian et al. [2017, 205] argue such approaches "will become a competitive necessity" as time progresses. Furthermore, Karl et al. [2016] argue that DevOps also applies to underlying infrastructure which includes hardware platforms, servers and so on. Therefore, a holistic view of DevOps should not limit it to just the development of software.

Taken together, Sebastian et al. [2017] suggest that both assets should overlap the chosen strategy. However, from a Software Engineering perspective it is difficult to perceive how both assets can work in harmony given Sebastian et al. [2017, 205] state that "some interviewees mentioned that their traditional Waterfall approach is evolving to a more collaborative, scaled down fast Waterfall". The notion of "fast Waterfall" as part of an Operational Backbone is especially confusing due to the non-iterative nature of such approaches when considering the iterative and incremental nature of Agile and DevOps. Nevertheless, Sebastian et al. [2017] also suggest regular software releases and Software as a Service (SaaS) as other possible delivery methods, which are arguably better aligned with Agile and DevOps [Chen, 2017; Gupta et al., 2017; Kneuper, 2017; Takimoto et al., 2016].

According to Takimoto et al. [2016, 8], DevOps forms an "integrated lifecycle from service planning and development to implementation and operation", which is continuous in nature (see figure 3.3). Such perspectives support the notion that "IT industry professionals work together on all phases of the life cycle of an information technology product, from design and testing to deployment and operation." [Veres et al., 2019, 106]

In this illustration, DevOps is shown as a iterative and functional harmonisation of software development and IT operations, where each works in tandem. This also suggests a collaboration between functions, highlighting where CD and CI fit into the cycle. Subsequently, neither function replaces the other, but rather adds support where capabilities, cultural and technological enablers are key to

realising DevOps [Smeds et al., 2015]. Taken together, Takimoto et al. [2016] and Smeds et al. [2015] offer a view that DevOps revolves around the transformation of core business systems, the utilisation of Agile processes and technical capabilities including, network abstraction and virtualisation; which also includes underlying infrastructure [Karl et al., 2016; Sill, 2015].

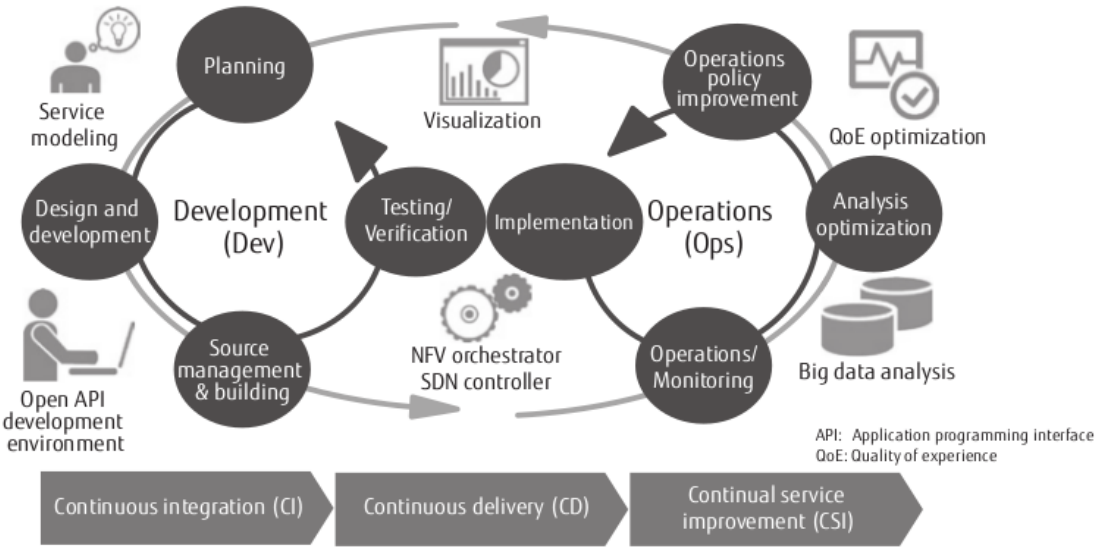


Figure 3.3: DevOps lifecycle outlined by Takimoto et al. [2016].

Another aspect of DevOps is that of various metrics and reporting which are argued to be a critical component [Dennehy and Conboy, 2017; Kim et al., 2016]. While important, metrics arising from DevOps practice are unique and as such may be difficult to repeat with machine learning techniques [Sun et al., 2016]. In the example set out by Takimoto et al. [2016], analysis of potentially large amounts of data should be continually undertaken, of which the output informs business process. Subsequently, Takimoto et al. [2016] argue that DevOps has the potential to generate big data, and should consider challenges such as how it is captured, stored and analysed, in order to better inform the business and thus enable a practice of continuous service improvement.

Tools have also received much research attention and are crucial enablers of DevOps [Airaj, 2017; Smeds et al., 2015; Wettinger et al., 2016]. Many conference and workshop papers are focused exclusively on DevOps tools, yet much of this

research lacks insight on their application in context [Dennehy and Conboy, 2017]. Numerous tools are available [Wettinger et al., 2017], however, many are proprietary rendering it difficult to integrate and combine with other tools being utilised [Wettinger et al., 2016]. Furthermore Wettinger et al. [2016] argue standards must play a greater role to tackle this problem, with the Topology and Orchestration Specification for Cloud Applications (TOSCA) being cited as one such standard to emerge. With DevOps becoming a preferred approach, Sill [2015, 74] argues that standards are necessary, especially to reduce the impact of complexity, and to "identify a common approach that can be reused". Additionally, Sill [2015, 72] asserts that the adoption of new standards is in itself a difficult endeavour, especially within a context of "rapid technological change". Some of the most commonly referenced tools include:

- Ansible
- AWS
- Azure
- Bitbucket
- Chef
- Digital Ocean
- Docker
- GitHub
- GitLab
- Hipchat
- Jenkins
- Jira
- Kubernetes
- Mercurial
- Logstash
- Prometheus
- Puppet
- Travis
- Trello
- Vagrant

While these tools are commonly used as examples, it is important to note that they are by no means exclusive to DevOps.

3.3.2.3 Cultural Implications

Much of the published literature on DevOps has taken a technical focus, overlooking the effects it has on culture. This is not surprising given most DevOps research is within the Computer Science and Software Engineering disciplines.

However, relatively recent research within the Business Management and Information Systems disciplines, focuses increased attention on the cultural impacts of DevOps adoption [Gupta et al., 2017; Sebastian et al., 2017].

As Agile practice and frameworks have evolved, software development has become an activity which is socially embedded given the increased emphasis on inter-team collaboration, shared values and knowledge [Dennehy and Conboy, 2017; Wettinger et al., 2017]. However, one of the biggest barriers facing organisations is that of organisational silos and that it is typical that "most IT services are organised in silos" [Airaj, 2017, 2]. Such silos have arisen from the 1990s whereby infrastructure was substantially different and where IT Operations handled contact with users of developed software [Roche, 2013].

While DevOps aims to bring newly developed software into production rapidly without sacrificing quality, Kneuper [2017, 79] argue that a "notoriously difficult interaction between development and operations" needs to be tackled. Furthermore, Wettinger et al. [2017, 282] argue that "collaboration is a key aspect for implementing DevOps practices and continuous delivery in particular because diverse experts such as developers and operations personnel are involved and thus need to collaborate".

Therefore a collaborative emphasis on process is required alongside shared goals in order to tackle the barrier between software development and IT operations [McLarnon et al., 2014; Wettinger et al., 2017]. However, such collaboration may challenge existing sociotechnical systems within the organisation, as McLarnon et al. [2014, 371] also argue that DevOps heralds a "paradigm shift that is changing how systems administration is viewed in relation to the other functions of a business, particularly in software development". Yet, software developers need to adapt as well, with Feitelson et al. [2013] highlighting how Facebook encourages a "perpetual development mindset", which is the practice of committing software regularly and often, but with increased emphasis on code quality.

Wettinger et al. [2016] argue that tools have emerged to attempt to help the cultural divide between Software Development and IT Operations. But tools are

just one part of a solution with many components, as Dennehy and Conboy [2017] argue that management should become familiar with the processes and tooling in use by others.

Moreover, Roche [2013] argues that the lack of universal definition for DevOps is aiding in polarising views on what it means. On one hand, DevOps is viewed as a job or role, which Roche [2013] argues is an adaptation from an existing role. Such a position fits with the notion of job crafting where employees reshape the boundaries of their jobs [Wrzesniewski and Dutton, 2001], which is overviewed in section 3.4. On the other hand, DevOps is a response to emerging needs in software development and support environments, which Roche [2013] claims is being driven by infrastructure advancements.

While DevOps maintains a key focus on software development, Fokaefs et al. [2017] argue that the cultural dimensions of DevOps take it beyond software development and IT operations teams, necessitating a strategic and cultural alignment from management. A similar argument is made from Fitzgerald and Stol [2017, 176] where they coin the term "BizDev" stating that "the link between business strategy and software development ought to be continuously assessed and improved". A similar term of "BizOps" is coined by Fokaefs et al. [2017], who defines it as the need for managers to align strategy, and preferably integrate the business with DevOps. Fokaefs et al. [2017] also argue that existing DevOps research is focused on software development activities, and therefore fails to consider the impact such activity can have on the business. However, Fokaefs et al.'s [2017] research is limited to software development activity within a cloud context and the authors have acknowledged their work has limitations which may hinder the generalisation of their findings beyond that context.

3.3.3 DevOps Research Agenda

DevOps is a relatively recent and fast moving phenomenon to emerge within both industrial and academic literature [Airaj, 2017; Fitzgerald and Stol, 2017; Kneuper, 2017; Roche, 2013].

Fitzgerald and Stol [2017] have identified a number of issues, and put forward a research agenda. This is presented where research questions need to consider three categories based on the views of stakeholders. These are: Business Strategy; Development; and Operations (see table 3.4).

Business Strategy	**Development**	**Operations**
Feature analytics	Continuous evolution and maintenance of software systems	Usage and prediction of product features
Continuous planning	Highly flexible architectures to enable continuous evolution	Sustaining customer trust in a product
BizDev	concurrent hardware and continuous software engineering and radical approaches to re-engineering a product	DevOps

Table 3.4: Research agenda adapted from Fitzgerald and Stol [2017, 187].

With feature analytics, Fitzgerald and Stol [2017] focus on the nature of information required by senior management for the planning and evaluation of features. This is necessary for software features to evolve and is informed by usage metrics. Continuous planning is inherent to the continuous delivery of product features [Takimoto et al., 2016; Wettinger et al., 2017], but specific focus is needed on how projects are aligned with business strategy and how software is designed, considering business requirements and how it will be continuously delivered while building and sustaining customer trust. Finally, Fitzgerald and Stol [2017, 176] put forward the notion of "BizDev", but argue there is an expectation mismatch between development and other areas of the business, which needs to be tackled. While Fitzgerald and Stol [2017] mention DevOps, it focuses on identifying barriers that prevent direct collaboration between Software Development and IT Operations. Nevertheless, when taken together with Wettinger et al.'s (2017) research, an argument can be put forward that collaboration must go beyond software development and IT operations, involve management, and that this collaboration is critical for DevOps.

3.3.4 Summary of Section

DevOps is an emergent topic of research, building on previous developments and theories in Software Engineering. However, DevOps is difficult to define, with polarised perspectives of it being either a concept or role. Multiple definitions exist, some of which are contradictory. Common themes of DevOps do appear around automation, change, collaboration, culture, process, and quality. Therefore this research will aim to identify a set of conceptual attributes and attempt to clarify a definition of DevOps, using the literature and input from practitioners.

While important for realising concepts of automation and CI, the heavy research focus on tools potentially overlooks wider implications of DevOps, which includes the strategic impacts it can have on an organisation. Although tools can be enablers, they alone are not DevOps.

Many organisations are reliant on legacy software systems which can have a multitude of maintainability issues. Many of these legacy systems are a potential barrier to DevOps adoption and due to the manner of processes within which they were developed, render them very difficult to bring into a DevOps approach. Furthermore, microservice software architectures are becoming increasingly popular and appear to fit well with DevOps. However, decomposing legacy software systems into microservices is a difficult undertaking.

As adopting DevOps can introduce significant scale of change for organisations, there is a critical need to link business, technical and software development strategies. This potentially means DevOps and organisational change are not mutually exclusive phenomena. As such it is important to also consider the wider sociocultural and sociotechnical implications that DevOps adoption could inherently introduce to an organisation. In conclusion, and despite the heavy technical focus on tools, it follows that DevOps is not exclusively a Computer Science and Software Engineering phenomenon. Rather, there is a substantial Business Management component that remains in great need of further empirical research. Therefore this PhD research will address the gaps in understanding DevOps adoption from a Business Management perspective.

3.4 Introduction to Job Crafting

The purpose of this section is to provide an overview of job crafting theory, which arose as a strong theme from abductive reasoning following a pilot of the diary study conducted for this PhD research. Additionally, the notion of work identity is introduced, defined and linked to job crafting. A summary is offered outlining the role job crafting will play as a theoretical lens for the case study of DevOps adoption undertaken for this PhD research.

Ilgen and Hollenbeck [1991, 173] state a job consists of a "set of task elements grouped together under one job title and designed to be performed by a single individual". In their seminal research, Wrzesniewski and Dutton [2001, 179] define job crafting as "the physical and cognitive changes individuals make in the task or relational boundaries of their work. Thus, job crafting is an action, and those who undertake it are job crafters". It follows, that tasks and relationships are critical to the employee-employer relationship, with job crafting focusing on an individual job holder's shaping of these boundaries. Wrzesniewski and Dutton [2001] propose that job crafting occurs at three levels: task, relationship and cognitive (see table 3.5).

Task job crafting occurs when an employee makes changes to their job's task boundaries. Wrzesniewski and Dutton [2001] argue that this happens through changes to the scope, type or quantity of job tasks employees choose to do, and in doing so, a different job is created to what was prescribed in the formal job specification.

Relationship is the second form of job crafting, where an employee changes how they interact with others in the workplace. Wrzesniewski and Dutton [2001] state that this form of job crafting involves employees choosing with whom and how frequently they want to interact with others which can also help determine the quality of any interaction. The change to the job occurs as a result of employees altering the nature of their relationships through changing their level of involvement with others at work [Wrzesniewski and Dutton, 2001].

The third form of job crafting proposed by Wrzesniewski and Dutton [2001] is cognitive, which occurs when the cognitive task boundaries of their job are changed by an employee. This can occur in many ways, but Wrzesniewski and Dutton [2001, 185] argue that it "likely involves employees' altering how they parse the job - viewing it either as a set of discrete work tasks or as an integrated whole". Such changes to how they perceive their job can radically and fundamentally change how an employee approaches it. Wrzesniewski and Dutton [2001] use the example of a nurse engaging in cognitive job crafting whereby they did not perceive their role to be just about delivering high-quality technical care, but rather about advocacy and providing holistic care for patients.

Job crafting	Description	Effect on Meaning of Work
Task	Changing the scope, type and quantity of job tasks	Work is efficient completed in a more timely fashion.
Relationship	Changing quality and/or amount of interaction with others encountered in the job	Job meaning changes so employees see their job as a vital part of an integrated whole.
Cognitive	Changing cognitive task boundaries, taking responsibility for information and insignificant tasks	Fundamental change to how an employee perceives and approaches their job.

Table 3.5: Task, relationship and cognitive job crafting, adapted from Wrzesniewski and Dutton [2001, 185].

Wrzesniewski and Dutton [2001, 186] argue that job crafting has the potential to shape an employee's "work identity", which refers to how an employee defines themself in the workplace. Through job crafting, employees take action to actively mould, shape and redefine their jobs, through the changing of task boundaries, adjusting the relationships between tasks and their colleagues, and changing their view of the job they do.

Figure 3.4 shows a model proposed by Wrzesniewski and Dutton [2001], showing five distinct stages of job crafting and illustrating the influences and effects it can have. Motivation is the first stage, where an employee is generally motivated by a need which may be work or socially derived.

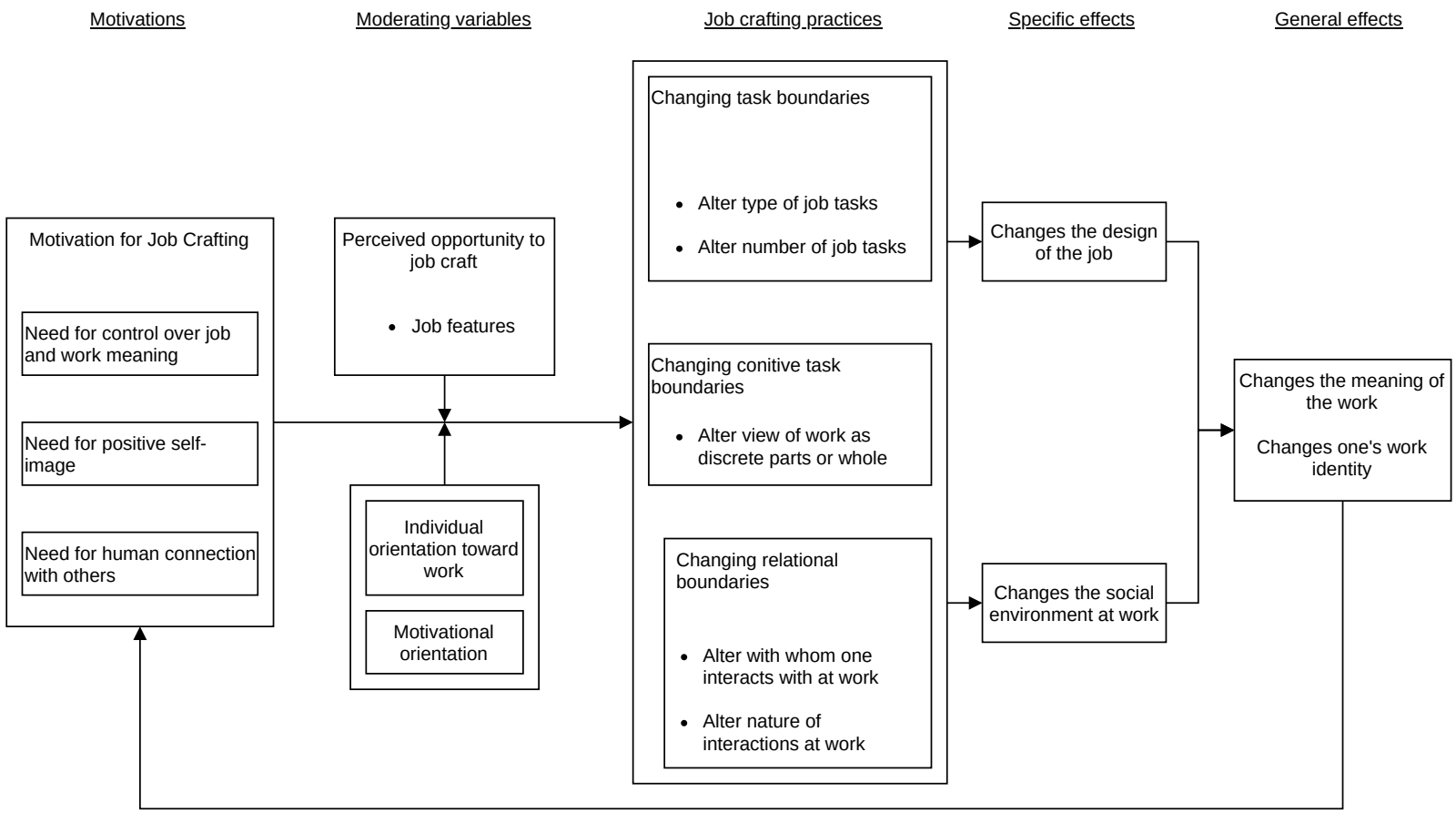


Figure 3.4: Model of job crafting showing how it can influence work identity, adapted from [Wrzesniewski and Dutton, 2001, 182].

With a motivation, an employee would then perceive what opportunity there is to actually job craft which in turn would be influenced by their intrinsic orientations toward their work. Wrzesniewski and Dutton [2001] argue job crafting is primarily an individual activity and an employee decides when and in what manner they reshape the boundaries of their job. Berg et al. [2010] argue that it is also a socially embedded phenomenon, focusing on an employee's perception of their position in the organisation's hierarchy. Berg et al. [2010] further argue this influences an employee's decision to job craft. Taken together, this example of employee perception would sit well with the moderating variables within Wrzesniewski and Dutton's [2001] job crafting model.

These moderating variables are followed by actual job crafting taking place, where the employee actively alters their tasks, relationships and view of their work. The specific effects of job crafting can be twofold, where an employee changes the design of their job and also the social environment at work. Finally, this can lead to more general and longer term effects such as what work means to the employee, and how they identify themselves at work. The model also presents job crafting as an iterative phenomenon.

3.4.1 DevOps and Job Crafting

There is little in depth research on job crafting in software engineering environments, but developers regularly face change due to innovation and technology evolution, which improves the environment within which they work [Chilton et al., 2005]. With DevOps being a recent phenomenon to emerge, there are no studies which apply job crafting within these environments.

What would motivate a software development or IT operations employee to job craft? In their study involving technology workers, including some software developers, Tims et al. [2014] linked employee self-efficacy with performance and in turn, a higher likelihood of job crafting. Furthermore, Mäkikangas et al. [2017] argue job crafting is not necessarily exclusive to individuals, but rather must be considered from an individual, team and organisational level.

However, Chilton et al. [2005] argue that the cognitive style of a software developer and perceived demand within this environment can affect their work identity. Mattarelli and Tagliaventi [2012] add that individual and collective job crafting may be as a response to change and perceptions of how employees identify themselves at work as a result. While job crafting may impact an employee's work identity, changes to it can also be a trigger for job crafting to occur [Wrzesniewski and Dutton, 2001]

Claps et al. [2015] identified 20 technical and social challenges an organisation can face when adopting continuous deployment (CD) (see figure 3.5). This may be relevant to consider for DevOps, as CD is one constituent of it [Chen, 2015; Claps et al., 2015], and also for job crafting given changing team roles and responsibilities were presented as a challenge, which could lead to employees crafting their jobs as a result [Mattarelli and Tagliaventi, 2012] and subsequently generating a new work identity [Wrzesniewski and Dutton, 2001].

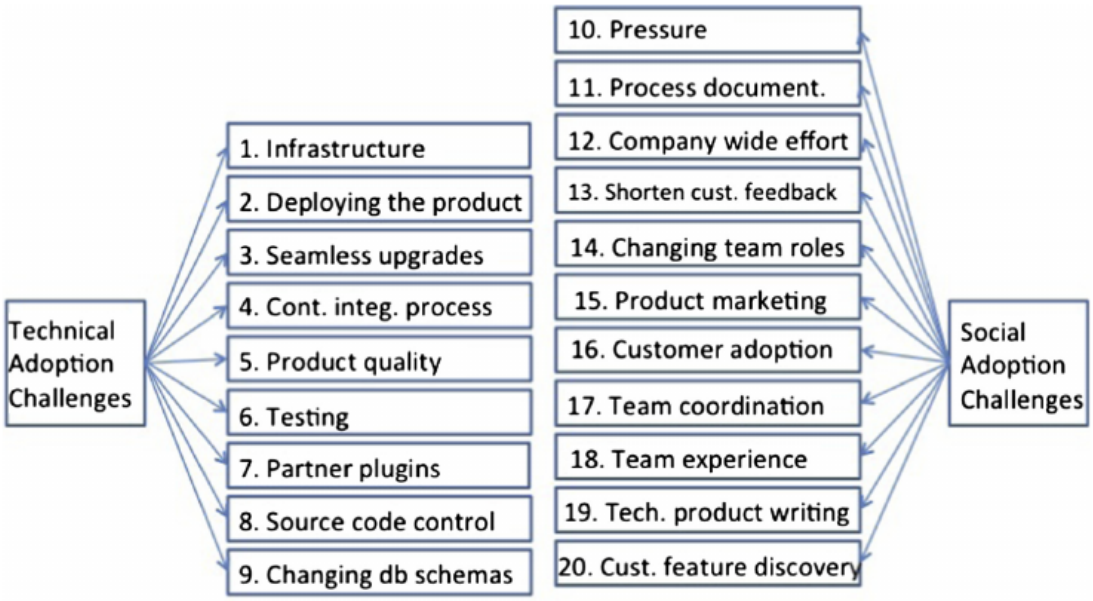


Figure 3.5: Technical and social adoption challenges when implementing the continuous deployment of software [Claps et al., 2015, 26].

Claps et al. [2015] argue that organisations need to be well prepared in order to manage the scale of change and challenges CD will introduce, both socially and

technically. This argument sits well with the proposition that DevOps accomplishes the transformation of software development through organisational change [Pass and Ronen, 2014]. Moreover, this change needs to be at the forefront of management and not merely viewed as an afterthought [Ghezzi, 2017]. The social challenges of changing team roles and team coordination add support to the argument that job crafting is socially embedded and that an employee's perception of where they fit into the organisational hierarchy can influence whether they job craft or not [Berg et al., 2010]. Sebastian et al. [2017] argue that management practices also needs to evolve in order to deal with software development related change [Sebastian et al., 2017].

Subsequently, managers should be included in any study concerning DevOps adoption as they too could engage in job crafting.

The agile approach to software development has helped keep pace with the demands of organisations and the increasing complexity of software. However, Agile has had limited focus on software development functions, failing to consider what happens with software after it has been developed [Gohil et al., 2011]. Gohil et al. [2011, 262] therefore argue that for an organisation to be "truly agile", the agile approach needs to move beyond software development to include other business functions, including IT Operations, in order to consider the various sub-systems of software, namely, the underlying infrastructure.

While Gohil et al. [2011] put forward the suggestion that agile practices are applied to IT operations, DevOps goes further with harmonising Software Development and IT Operations functions [Allman, 2012; Liu et al., 2014; Loukides, 2012; Mohamed, 2015; Tamburri et al., 2015]. Either approach would necessitate substantial change, but DevOps potentially reshapes the work environment with a new culture emerging as new relationships are built [Walls, 2013]. It follows that work identities may also change as a result [Wrzesniewski and Dutton, 2001].

Despite some research on job crafting in Software Development, further exploratory work is needed to better understand how employees engage in job crafting activity, especially with the continued evolution of process and technology.

3.4.2 Summary of Section

Job crafting was theorised by Wrzesniewski and Dutton [2001] to explain how, why and in what manner an employee actively makes changes to their job. Three forms of job crafting are proposed: task; relationship; and cognitive. The concept of work identity is also put forward to describe how an employee identifies themself in their place of work. Wrzesniewski and Dutton [2001] argue that an employee's work identity can be changed as a result of job crafting.

Although widely studied, job crafting has seen limited application in the context of software development. This research will utilise job crafting as a theoretical lens to understand how and why IT Operations employees and software developers change their job boundaries as DevOps is adopted by the organisation. Finally, Berg et al. [2010] argue job crafting is socially embedded which potentially links to the social challenges put forward by Claps et al. [2015] (2015), meaning DevOps therefore has a management component. This PhD research will include managers and also explore the management implications of DevOps.

3.5 Summary of Literature Review

This chapter had three aims: Firstly, to provide a narrative overview of how software and software engineering emerged, and to set the context for DevOps. Secondly, to provide a systematic review of current DevOps research and identify any problems and deficiencies with it. This was accomplished by following the protocol outlined in section 2.6. Thirdly and finally, due to the abductive reasoning from the pilot study (see section 2.5.2), the theory of job crafting was introduced and overviewed for its application to this research and software engineering in general.

Emerging as a discipline during the 20th century, Software Engineering has roots in the first examples of programming from the late 18th and 19th centuries, with early pioneers including Joseph Marie Jacquard (1752-1834) and Ada Lovelace

(1815-1852). Although a comparatively young discipline, Software Engineering has made some major contributions over the last century. Early approaches to developing software were lacking in structure and method, leading to the 'software crisis' [Randell, 1996] and the subsequent emergence of the first structured approaches. Software development continued to evolve with the emergence of iterative and incremental approaches, eventually formalised as Agile [Beck et al., 2001]. The emergence of DevOps heralds a potential paradigm shift in process as it extends beyond software development activity to include IT operations.

The systematic review of the DevOps literature presents DevOps as an emerging and growing field of research from 2010. However, much of the literature is dominated by a focus on tools, rather than a holistic view of how DevOps is adopted and affects organisations and those working in Software Engineering roles.

While many definitions have been put forward, DevOps is difficult to define, with inconsistent, conflicted and polarised definitions [Dyck et al., 2015; Roche, 2013; Smeds et al., 2015]. While there appears to be general agreement that DevOps can harmonise two traditional organisational silos, the literature also suggests DevOps extends the Agile approach beyond software engineering, to include IT operations. The literature especially highlights the practices of CI and CD to be intrinsic to DevOps, which would inherently involve both software development and IT operations functions. Furthermore, Takimoto et al. [2016] present a DevOps lifecycle model, showing DevOps to be iterative in nature.

Moreover, it is argued that organisational change is inherent to DevOps [Pass and Ronen, 2014], but this is often viewed as an afterthought [Ghezzi, 2017]. Sebastian et al. [2017] argue that managers must be proactive in dealing with such change due to the impact it can have on the organisation, particularly where software is critical to its operation.

However, despite the frequency of peer-reviewed literature on DevOps increasing over the decade, deficiencies remain in interdisciplinary empirical research activity, particularly with regards to what DevOps means to an organisation.

Job crafting is a theory to explain how and why employees make changes to their jobs [Wrzesniewski and Dutton, 2001]. Although there is some limited job crafting work with software developers [Tims et al., 2014], no studies have explored job crafting and DevOps. Taken together with the argument that software developers and IT professionals regularly face change in their roles [Chilton et al., 2005], job crafting offers a useful theoretical lens given the inherent impacts DevOps could have on employees and organisational culture.

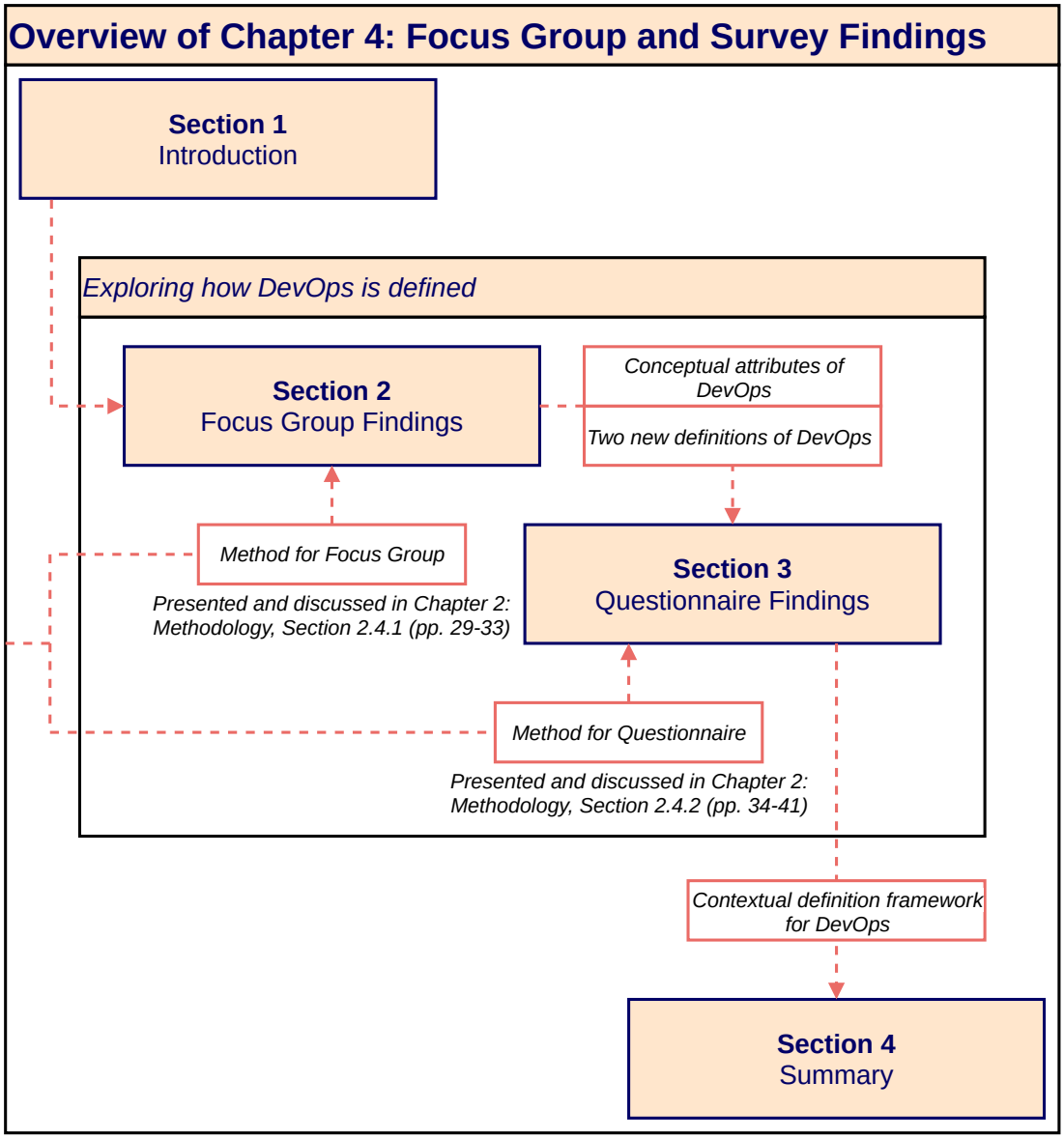
Therefore, taking together the findings within the literature review and the pilot study, this PhD research will explore how DevOps can be defined and undertake a longitudinal empirical study in order to explore why an organisation would adopt DevOps. The research will seek to identify what the perceived benefits of DevOps adoption are, and if they are realised. As the literature has highlighted links between DevOps and organisational change, this research will also seek to explore the extent of any DevOps driven change. Finally, job crafting will be applied as a theoretical lens to understand the multi-faceted nature of the changes and effects that DevOps practice has on employees.

Chapter 4

Focus Group and Survey Findings

"Currently, DevOps is more like a philosophical movement, not yet a precise collection of practices, descriptive or prescriptive."

– Gene Kim


Overview of Chapter 4: Focus Group and Survey Findings
Section 1
Introduction
Exploring how DevOps is defined
Section 2
Focus Group Findings
Conceptual attributes of DevOps
Two new definitions of DevOps
Method for Focus Group
Presented and discussed in Chapter 2: Methodology, Section 2.4.1 (pp. 29-33)
Section 3
Questionnaire Findings
Method for Questionnaire
Presented and discussed in Chapter 2: Methodology, Section 2.4.2 (pp. 34-41)
Contextual definition framework for DevOps
Section 4
Summary

4.1 Introduction

This chapter presents the findings from a focus group and questionnaire, which explored the definition of DevOps. The outputs from the focus group were a set of 17 conceptual attributes and two definitions of DevOps based on them. This output was also tested in a questionnaire survey to gauge wider agreement and to help inform a 14 month case study of DevOps adoption.

4.2 Focus Group Findings

The focus group was structured into two exercises, the first of which participants discussed what they would consider to be attributes of DevOps. In the second exercise, participants were asked to create two definitions; one from scratch and the other derived from existing definitions in the literature.

The literature shows that DevOps appears difficult to universally define (see section 3.3). In exploring this, the first focus group exercise was to identify and agree on a set of conceptual attributes of DevOps. Participants worked in two groups for this, as described in section 2.4.1.

Exercise two once again had participants working in one of two groups, this time to attempt to define DevOps using the attributes agreed in exercise one as a guide. One definition was created from scratch, while the other was derived from definitions found within peer-reviewed literature.

4.2.1 Framework for Contextually Defining DevOps

The output from exercise one of the focus group was a set of 17 conceptual attributes of DevOps. A consensus between the participants was found, resulting in the attributes being agreed and placed within four groups (see table 4.1 and figure 4.1).

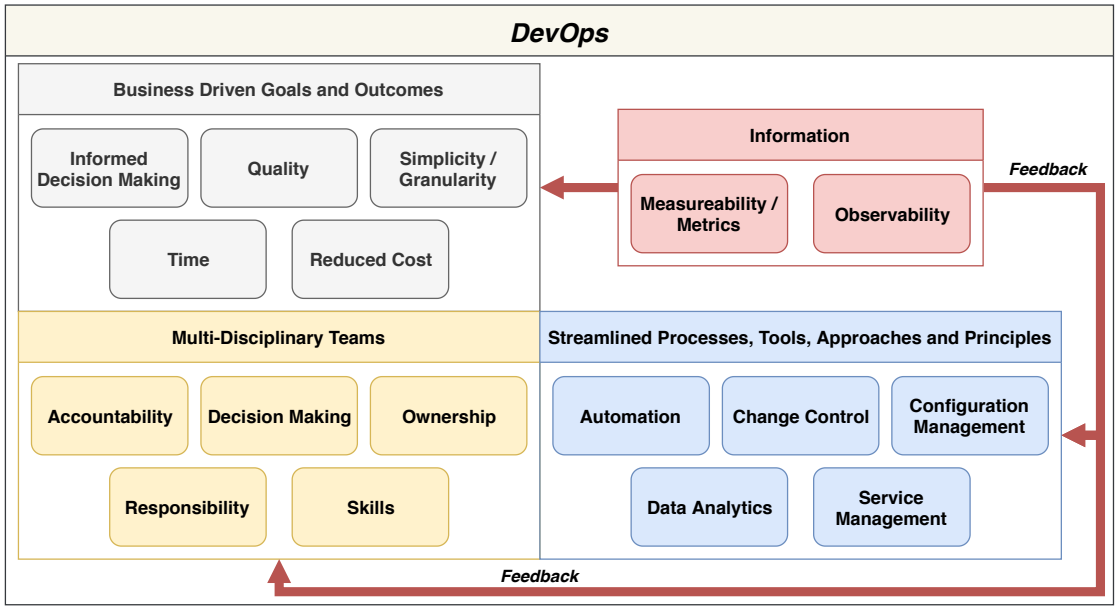


Figure 4.1: Conceptual attribute framework for DevOps

The focus group participants discussed and appraised the attributes, placing them into four distinct groups.

The streamlined processes, tools, approaches and principles group includes the conceptual attributes of Automation, Change Control, Configuration Management, Data Analytics and Service Management. Participants agreed these are critical for organisations to consider when adopting DevOps. Furthermore, continuous integration and deployment are considered within the Automation attribute.

Multi-disciplinary teams, focuses on both individual employees and teams. However teams are structured in an organisation, they must operate as a collaborating multi-disciplinary unit, with shared ownership and mutual accountability. Participants also agreed that Decision Making, Responsibility and Skills are crucial to consider for any team engaging in DevOps.

Finally, participants agree that feedback and learning is key to the successful adoption of DevOps. Therefore, the fourth category was Information and contained the attributes of Observability and Measureability/Metrics.

Continuous feedback is critical to DevOps with a feedback arrow shown between the groups of attributes so as to enable continuous improvement and allow the DevOps process to evolve.

The grouped conceptual attributes and feedback loop of DevOps offers a framework for managers to help holistically define DevOps but within the context it is adopted. A crucial finding, which is also shown in the framework, was the agreement that DevOps should be considered and driven from a strategic level. Therefore, the business driven goals and outcomes group includes Informed Decision Making, Quality, Reduced Cost, Simplicity/Granularity and Time.

Finally, the framework was then utilised by the focus group participants to produce two new definitions of DevOps; one from scratch and the other derived from nine existing definitions in the literature. The final focus group activity was an evaluation and appraisal of both definitions, which is presented in section 4.2.2.

Group	Attribute	Description
Streamlined Processes, Tools, Approaches and Principles	Automation	Automation of testing, deployment and infrastructure provision.
	Change Control	Controlling the pace of software and organisational change.
	Configuration Management	Configuration and provisioning of necessary IT infrastructure.
	Data Analytics	The analysis of performance and run time data.
	Service Management	Seamless management of software and support services.
Multi-Disciplinary Teams	Accountability	Mutual accountability to colleagues, including managers.
	Decision Making	Joined up thinking and team owned decisions.
	Ownership	Employees taking ownership of software development and release.
	Responsibility	Promotion and emphasis of shared responsibility across roles.
	Skills	Enabling individuals to acquire new knowledge and skills.
Business Driven Goals and Outcomes	Informed Decision Making	Strategic decisions being informed by continuous feedback.
	Quality	Quality of both developed software products and processes.
	Reduced Cost	Value improvement by delivering working software sooner.
	Simplicity/Granularity	Promotion of leaner, streamlined and more efficient processes.
	Time	Reducing lead time for software to be released to users
Information	Measurability/Metrics	Data concerning software, process, performance and usage.
	Observability	Transparency of software engineering and operating processes.

Table 4.1: Conceptual attributes of DevOps with descriptions, as agreed by the focus group participants.

4.2.1.1 Focus Group Definition One - From Scratch

The first definition was produced from scratch, but utilised the conceptual attributes as a guide.

"*DevOps is a continuous improvement methodology that uses a set of tools, streamlined and automated processes, and empowered, multi-disciplinary teams to deliver, operate and inform business outcomes.*"

With definition one, participants placed great emphasis on DevOps ultimately having business driven goals and outcomes, therefore going beyond tools. This also opens up the cultural and soft-skills aspect that comes with people. Participants also agree that teams should be multi-disciplinary in a DevOps environment with management enabling them to develop necessary skills and culture.

Finally, and most critical, is the emphasis on a continuous feedback loop based on measurement and observability. Participants agreed that DevOps has the potential to unlock a wealth of data which serves not only to provide feedback at a strategic level, but can also enable continuous improvement. All participants agree that continuous feedback is critical to the success of DevOps in any setting.

4.2.1.2 Focus Group Definition Two - Literature Derived

Unlike the first, the second definition was derived from those put forward in peer-reviewed literature. The participants were provided with nine definitions (see table 3.1), which were discovered through a systematic review of the DevOps literature. Participants were instructed to evaluate each definition and ultimately produce a new one from these while utilising the previously agreed conceptual attribute framework as a guide. A new definition of DevOps derived from the work of Bass et al. [2015], Mohamed [2015] and Dyck et al. [2015] (see table 4.2) was proposed:

"*DevOps is an evolution in how IT services are delivered and supported. It stresses cross functional collaboration to bridge the organisational process divide between*

development and operational teams. It aims to reduce the time between committing a change to a system and the change being placed into production."

Definition
Two: "A set of practices intended to reduce the time between committing a change to a system and the change being placed into normal production, while ensuring high quality" [Bass et al., 2015, 4]
Eight: "DevOps is an evolution in thinking with regards how IT services are delivered and supported. It is a continuation of some of the predecessor work in the areas of continuous integration and application life cycle management (ALM); therefore, it is rooted in the agile philosophy, which also attempts to bridge the traditional organizational process divide between development and operations teams" [Mohamed, 2015, 51]
Nine: "DevOps is an organizational approach that stresses empathy and cross-functional collaboration within and between teams - especially development and IT operations - in software development organizations, in order to operate resilient systems and accelerate delivery of changes" [Dyck et al., 2015, 3]

Table 4.2: Participant selected definitions of DevOps from the literature.

In evaluating literature definition two [Bass et al., 2015], participants agreed that it offered a good abstract point of view. In particular, no focus on specific roles or methods, but instead emphasising the speed of getting commits into production was welcomed. Additionally, participants perceived the definition implied software development and IT Operations as functions rather than specific teams. Quality was considered a positive, however, participants did report the term "high quality" was ambiguous.

Definition eight's [Mohamed, 2015, 51] statement of DevOps as an "evolution in thinking with regards to how IT services are delivered and supported" was especially well received by participants as it alludes to change in both software development and IT operations functions. Further more, the evolutionary meaning from this definition sits well with how DevOps can tackle traditional silos of both functions. While positively received by participants, definition eight was deemed to over-focus on specific approaches. Moreover, it was agreed that DevOps is inherently an Agile approach and therefore definition eight need not mention this explicitly.

Definition nine's [Dyck et al., 2015] focus on culture, people and teams was positively received by the participants. Positive comparisons were drawn with the definition offered by Bass et al. [2015], where both appear complimentary. Furthermore, participants agreed with Dyck et al.'s (2015) reference to promoting empathy.

All remaining definitions were dismissed, as shown in table 4.3.

Definition
One: "The 'DevOps' approach to system administration introduces best practises from software engineering" [Obstfeld et al., 2014] **Comments**: Limited and too vague to glean any meaning.
Three: "DevOps is a movement within software engineering that professes to bring software developers and operations staff (those in charge of infrastructure, quality control, packaging, and release of software products) in close alignment, to ensure harmonious tasking and smooth transition of project artefacts through interoperable processes and tools" [Cois et al., 2014] **Comments**: Good it alludes to change, but takes a narrow view and is too verbose to glean any additional meaning.
Four: "The DevOps movement addresses the gap between developers and operational teams in enterprise networks by borrowing techniques from agile programming practices, building tools that automate well-known manual steps" [Császár et al., 2013] **Comments**: Reference to process and automation are positive. However, there are no goals or drivers and seems to imply there is always a 'gap'. Finally, the term "borrowing" purports DevOps is not agile.
Five: "A cultural movement combined with a number of software related practices that enable rapid development" [Walls, 2013] **Comments**: Cultural view of DevOps is interesting, but otherwise the definition is very limited and vague.
Six: "A set of engineering process capabilities supported by certain cultural and technological enablers" [Smeds et al., 2015] **Comments**: Vague and with very little meaning.
Seven: "A practice aimed at repairing the schism between the two teams." [Hosono, 2012] **Comments**: Limited and negative assertion of a "schism" between teams, which is not always the case.

Table 4.3: Literature definitions of DevOps dismissed by participants.

4.2.2 Focus Group Evaluation of Agreed Definitions

With the two definitions created, the focus group participants re-convened as one group to evaluate both. Starting with definition one, participants praised its business focus with wide agreement that DevOps needs to be driven by business goals and outcomes. Secondly, the emphasis on empowering a multi-disciplinary team was well regarded by participants.

However, some participants felt that including 'tools' in this definition detracts from its value. Additionally, the overlap with culture was too implicit, with participants agreeing that culture should be more explicitly visible within the definition.

Definition two's use of the term 'evolution' was positively received participants. They felt this helped showcase DevOps as something that continually changes, therefore illustrating the inherent transitional context of DevOps. Additionally, participants felt that empathy was implied as a key element to facilitate 'cross-functional collaboration'.

While the second definition was received as well as the first, participants agreed the definition contained a negative connotation of the word 'divide'. Therefore they felt it asserted barriers were always present between software development and IT operations functions.

Finally, and most critically, all participants of the focus group agreed that DevOps is, universally at least, very difficult to define, thus concurring with previous literature in defining DevOps [Dyck et al., 2015]. However, the participants expressed that DevOps realisation is ultimately going to differ between organisations. Therefore, the focus group concluded that using the conceptual attribute framework as a guide (see figure 4.1), managers, IT Operations professionals and software developers can define DevOps in the context of its adoption rather than rely on any attempted universal definition.

4.3 Questionnaire Findings

The outputs of the focus group, namely the conceptual attributes and two definitions, were placed into a questionnaire survey which was completed by a total of 83 anonymous respondents within the wider DevOps community.

Each respondent was asked to state their agreement on each conceptual attribute previously agreed, in addition to evaluating the two definitions produced in the focus group.

4.3.1 Conceptual Attributes - Exploratory Factor Analysis

Analysis of the questionnaire data revealed that the responses were spread with none of the focus group defined conceptual attribute groups loading well. Subsequently, each conceptual attribute was considered separately through an Exploratory Factor Analysis (EFA) to identify any latent relationships within the data.

One such factor was found with 'Decision Making', 'Ownership' and 'Responsibility'. While all three of these conceptual attributes were part of the "Team" group in the framework, they loaded well when not including 'Accountability' and 'Skills'. As a result, the factor was simply defined as Team.

A Cronbach's Alpha test was performed to validate the resulting model producing an α value of 0.76 and average variance of 0.53.

The Team model was then used as a predictor with each of the remaining attributes in a regression test. This produced two further results showing "Measurability/Metrics" and "Accountability" are influenced by the team factor (see figure 4.2).

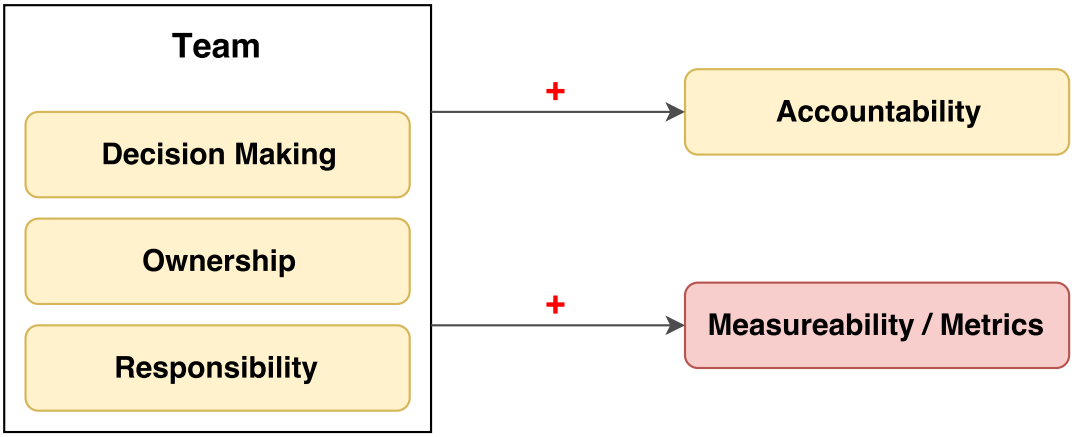


Figure 4.2: Model of the team factor of DevOps conceptual attributes showing the positive influence on Accountability and Measureability/Metrics.

4.3.2 Conceptual Attributes - Inter-rater Agreement

While the previous analysis revealed four team attributes to be important, this does not suggest the others should be dismissed. To examine these further, the inter-rater agreement of the attributes using Cohen's Weighted Kappa (κ_w) was sought.

As outlined in the earlier methodology chapter, the κ_w statistic is being used to determine agreement by respondent domicile, in this case, between UK and Non-UK based respondents. The value of κ_w is interpreted using the strengths of agreement outlined by Landis and Koch [1977].

Overall, there is generally good agreement with the conceptual attributes. However, Simplicity/Granularity, Automation, Change Control, Reduced Cost, Service Management and Observability showed statistical significance with moderate or substantial strengths of agreement (see table 4.4), thus the null proposition is rejected.

Attribute	$\mathcal{K}_w$	Strength	Sig.
Simplicity/Granularity	0.648	Substantial	*
Automation	0.603	Substantial	*
Change Control	0.681	Substantial	*
Reduced Cost	0.521	Moderate	*
Service Management	0.507	Moderate	*
Observability	0.483	Moderate	*
Quality	0.468	Moderate	N.S
Ownership	0.459	Moderate	N.S
Measurability/Metrics	0.375	Fair	N.S
Responsibility	0.375	Fair	N.S
Informed Decision Making	0.366	Fair	N.S
Accountability	0.34	Fair	N.S
Skills	0.308	Fair	N.S
Time	0.268	Fair	N.S
Decision Making	0.268	Fair	N.S
Data Analytics	0.110	Slight	N.S
Configuration Management	0.068	Slight	N.S

p * < 0.05, ** < 0.01, *** < 0.001

Table 4.4: $\mathcal{K}_w$ values on attributes between UK and Non-UK respondents with strength according to Landis and Koch [1977].

4.3.3 Evaluation of Focus Group Produced Definitions

Respondents were asked to specify a preference for either the first or second definition previously created in the focus group. The second definition, which was derived from the literature was preferred by a greater number of respondents (see table 4.5). However a χ^2 value of 0.766 was calculated, where p = 0.38. As such there appears to be no significance over definition preference.

Definition	Respondents	%
One (produced from scratch)	34	41%
Two (derived from literature)	49	59%

Table 4.5: Questionnaire respondent preference on focus group produced definitions

4.3.2.1 Themes Derived from Definitions

Respondents were also asked to provide positive and negative comments for each definition. Given the varying nature of the qualitative responses, they were interpreted and coded into a single word or phrase, with a complete list of themes from each definition provided in appendix 10 on page 238. In both definitions respondents were positive towards the focus on team and culture. Conversely, it was felt both definitions were limited, failing to capture the full essence of what respondents believe DevOps is; and contained buzzwords throughout. Thus these results further echo the focus group findings that DevOps is better defined in context. Figures 4.3 and 4.4 provide an overview of coded positive and negative theme frequency for each definition.

Definition one's focus on Automation and Multi-Disciplinary Teams was received well by respondents. While business outcomes were considered positive, there was an almost equal frequency of respondents feeling this was also a negative. Finally, respondents liked the concise, simple and succinct wording this definition provided.

While lacking the focus on automation, respondents felt definition two was much more focused on collaboration, especially between different functions. In addition, there was a positive response to DevOps being about time reduction, process and delivery. Moreover, the idea that DevOps is an 'evolutionary' concept sat well with some respondents. Definition two was also criticised for being more verbose and perceived as 'academic' to most respondents, as well as lacking any focus on automation. The notion of DevOps as an 'evolution' was positively received by some respondents, but others viewed it negatively too.

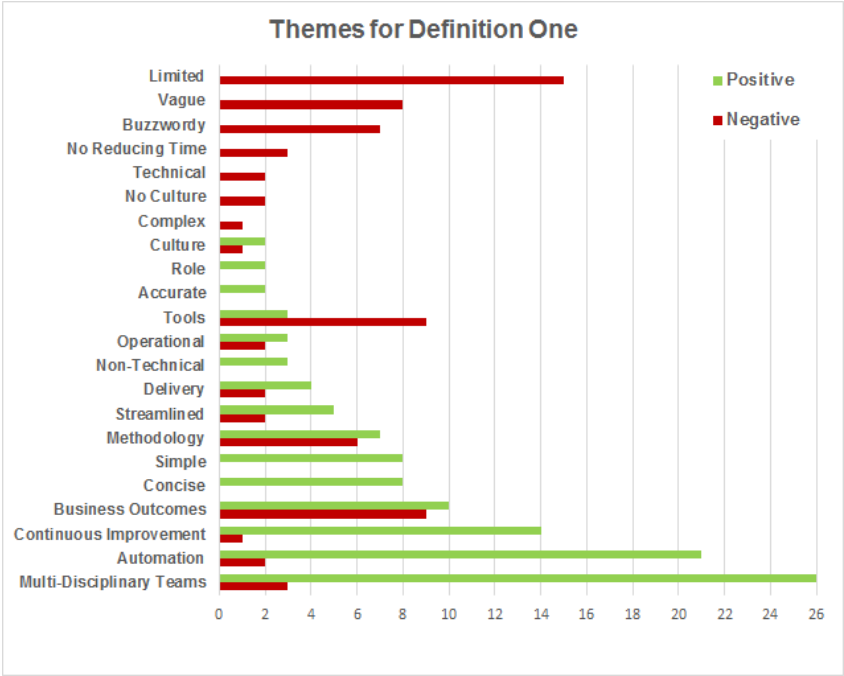


Figure 4.3: Frequency of positive and negative themes for definition one.

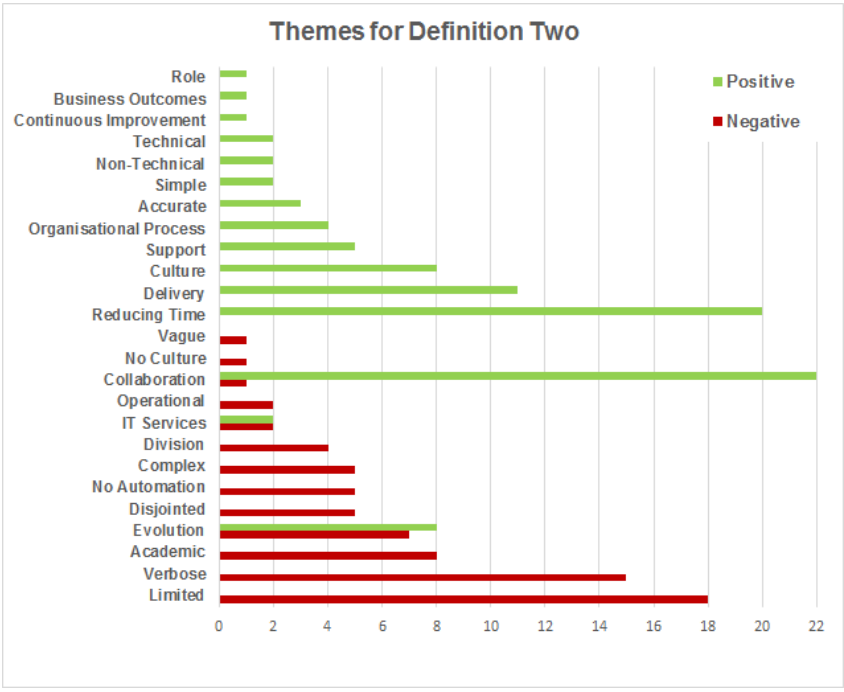


Figure 4.4: Frequency of positive and negative themes for definition two.

4.4 Summary of Focus Group and Questionnaire Findings

In this chapter, the results of a focus group and questionnaire survey exploring the definition of DevOps were presented.

The focus group identified and agreed a set of 17 conceptual attributes, offering a framework for contextual definitions of DevOps. Furthermore, two definitions were proposed, one designed from scratch and the other derived from existing literature definitions. Focus group participants did however agree that DevOps is difficult to define, at least universally.

However, the survey responses showed no distinct preference for either definition with respondents providing different positive and negative feedback on both. With the DevOps attributes, the inter-rater agreement results revealed a good overall strength of agreement. Additionally, an exploratory factor analysis revealed a factor with the questionnaire responses concerning the team grouping of conceptual attributes. Following further analysis, and confirmation of the model, these results suggest decision making, ownership and responsibility within the team have a positive influence upon accountability and measurability/metrics in a DevOps context.

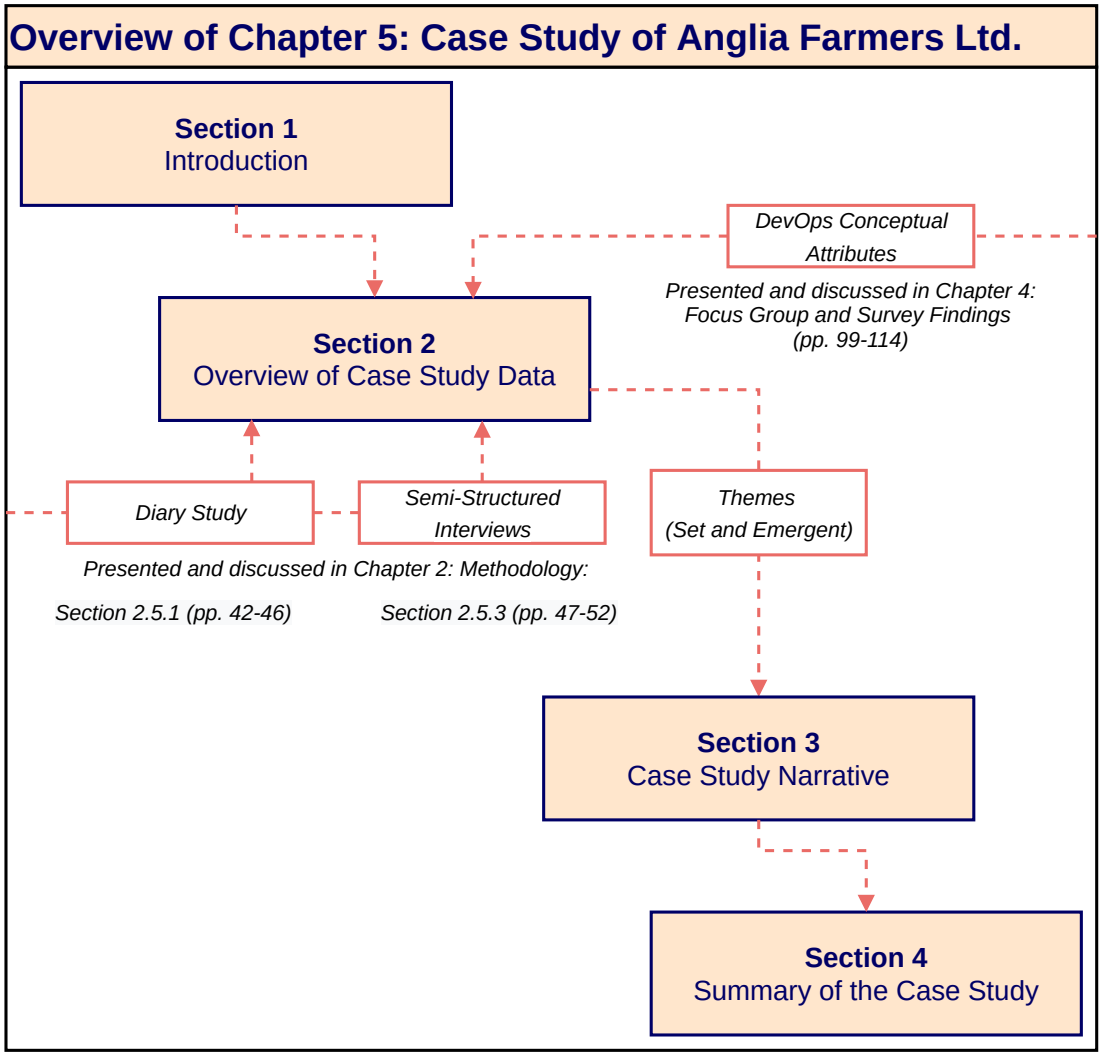
Analysis and findings from the focus group and questionnaire data on DevOps definitions were used to help design the case study presented in chapter 5. The conceptual attributes of Simplicity/Granularity, Automation, Change Control, Reduced Cost, Service Management and Observability showed statistical significance and were used to fine tune the thematic analysis undertaken for the qualitative case study of DevOps adoption at Anglia Farmers Ltd. This was especially helpful in guiding focus on the non-technical themes that emerged as important through this part of the research.

Chapter 5

Case Study of Anglia Farmers Ltd.

"Case studies are analyses of persons, events, decisions, periods, projects, policies, institutions, or other systems that are studied holistically by one or more methods."

– Gary Thomas


Overview of Chapter 5: Case Study of Anglia Farmers Ltd.
Section 1
Introduction
DevOps Conceptual Attributes
Presented and discussed in Chapter 4: Focus Group and Survey Findings (pp. 99-114)
Section 2
Overview of Case Study Data
Diary Study
Semi-Structured Interviews
Themes (Set and Emergent)
Presented and discussed in Chapter 2: Methodology:
Section 2.5.1 (pp. 42-46)
Section 2.5.3 (pp. 47-52)
Section 3
Case Study Narrative
Section 4
Summary of the Case Study

5.1 Case Study Introduction and Overview

In this chapter, a case study is presented which follows DevOps adoption at Anglia Farmers Ltd (AF). AF is the UK's largest agricultural purchasing group with over 150 employees, thus falling into the medium sized business category according to Rhodes [2016].

Figure 5.1: AF's logo and offices in Honingham Thorpe, Norfolk

At the core of AF's business operations is a software system called AFI. AFI is used by AF as an information system to record details of its customers, purchases and invoices. Additionally, customers can login to the system themselves to view this information as well as any relevant industrial information aggregated by AFI.

AFI is over a decade old and its development was outsourced. Yet, despite being legacy software, AFI is a critical component of AF's operation. AF maintained a single software developer for the provision of localised software maintenance, although most development work was outsourced. In addition, AF has an 'IT Operations and Support' team comprising two systems administrators who provide and maintain the infrastructure needed for hosting AF's software as well as day-to-day end-user IT support for AF's employees.

Despite being critical to the continuity of business for AF, AFI is considered "no longer fit for purpose" by senior management. Therefore, AF's senior managers took the decision to develop a replacement for AFI, but were keen to ensure the quality of the replacement system was of a much higher standard. This was followed up with a decision to develop the replacement internally, resulting in AF appointing a Software Development Manager, with experience of Agile software development. Subsequently AF initially employed an additional four software

developers and one test analyst to develop "Harrier", the intended replacement system for AFI.

The software development manager first explored DevOps in 2015 with a view to enhancing development practice at AF. Of particular attraction was the continuous deployment and release to the business of developed Harrier features.

5.1.1 Justification for Case Study Selection

AF were selected for the case study of DevOps adoption because of their decision to insource software development activity for the new Harrier system. While DevOps was not initially considered, the software development manager appointed by AF intended to adopt a DevOps approach to Harrier's development following their own research into it. This in turn links AF as an appropriate case study organisation for answering research questions 2, 3 and 4 (see section 1.3). Furthermore, there was a large degree of convenience given AF are based locally to the researcher and that AF and all participants were happy to commit to 14 months of study.

5.1.2 Structure of Case Study

An overview of the qualitative data and thematic analysis undertaken is provided in section 5.1.3, before the main case study narrative begins in section 5.2.

This case study follows DevOps adoption at AF over 14 months from January 2016 to March 2017 inclusive, with the narrative presented over three distinct 'time periods' as illustrated in figure 5.2. These were derived following the scheduled semi-structured interviews which occurred in January 2016, May 2016, December 2016 and March 2017. Quotes are taken and presented from the respective time period within the case study narrative. Additionally, a reference to the qualitative data is given, with a prefix of I or D, denoting it being from a diary or interview respectively.

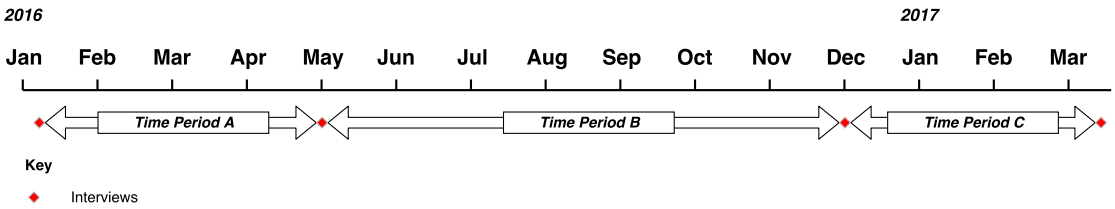


Figure 5.2: Structure and timeframe of the AF case study.

The case study's phenomenon in context is DevOps with the boundary set to the Software Development (Dev), IT Operations and Support (Ops) and related management functions at AF. It does not consider other departments at AF other than where they have been explicitly mentioned by participants. Therefore, the case study maintains focus on exploring DevOps adoption at AF while considering the wider activities of Dev, Ops and management.

While AF have agreed to be named in this thesis, each participant in the study has been anonymised to retain their confidentiality. Therefore where participants are mentioned, they are referred to as 'P1', 'P2' etc. An index of quotes taken from the qualitative case study data is provided in appendix 12 on page 242.

5.1.3 Overview of Case Study Data

A thematic analysis was conducted on the qualitative data acquired from 13 participants over a 14 month period of study at AF. The data were collected through open reflection diaries and a series of semi-structured interviews.

Eight set-themes were previously defined following the earlier analysis of focus group and questionnaire data concerning the conceptual attribute framework. In particular, themes were set from the conceptual attributes following the questionnaire inter-rater agreement analysis. Additionally, task, relationship and cognitive job crafting were defined as set-themes given the abductive reasoning which identified these during the pilot diary study.

Moreover, as this research is exploratory in purpose, it considered a total of 24

themes during the study:

- Automation
- Accountability
- Business Management
- Cognitive Crafting
- Collaboration
- Continuous Integration
- Control
- Culture
- Decision Making
- Knowledge Management
- Legacy Systems
- Measurability / Metrics
- Ownership
- Planning
- Process
- Quality
- Relationship Crafting
- Release
- Resistance
- Responsibility
- Task Crafting
- Technical Debt
- Transformation
- Work Identity

Through the diary study and semi-structured interviews, a large quantity of qualitative data were collected and analysed according to the method put forward in section 2.5.4. Themes were coded from the data in a manner that captured the time, participant and any secondary, tertiary or overlapping themes, therefore taking into account the sociocultural and sociotechnical context. For each theme, raw text was extracted from transcripts and a summary provided. A specimen example of these coded themes can be found in appendix 11 on page 240.

Given the large quantity of data, three rounds of coding were undertaken with the aim of consolidating the data. The first round resulted in a total of 609 theme instances comprised of 211 diary and 398 interview themes.

The data were then consolidated in the second round, where duplicate and over-

lapping theme instances were addressed through merging. At this point the 'Technical Debt' theme was discarded as it had few instances which when analysed and interpreted offered no specific insight or value to the study. By the end of the second round, a total of 442 theme instances were recorded, comprising 170 diary and 272 interview themes.

The final round of coding further consolidated the data, with 12 themes re-coded or discarded entirely (see table 5.1). The final result of the data coding was a total of 415 recorded themes, comprising 153 from diaries and 262 from interviews.

Theme	**Description**
Accountability	Merged with Responsibility. Both used interchangeably.
Automation	Occurred as a secondary or tertiary theme.
Control	Merged with Ownership.
Task Crafting	Abstracted within a new primary theme called 'Job Crafting'. Each type was recorded as a secondary or tertiary theme accordingly.
Relationship Crafting	
Cognitive Crafting	
Measurability/Metrics	Occurred as a secondary or tertiary theme.
Planning	Discarded due to low frequency always secondary or tertiary to Culture.
Quality	Occurred as a secondary or tertiary theme.
Resistance	Occurred as a secondary or tertiary theme.
Technical Debt	Discarded given low frequency with little value.
Transformation	Occurred as a secondary or tertiary theme.

Table 5.1: Third round theme coding, consolidation and merging

Figure 5.3 illustrates the final frequency of themes coded from the data. Some themes were more apparent depending on the data collection method. For instance, the themes of Culture, Process and Work Identity had greater prominence from interview data whereas Collaboration, Ownership and Decision Making were more apparent from diary data. Furthermore, Job Crafting, while slightly more frequent in diary data, is close to parity with both data collection methods.

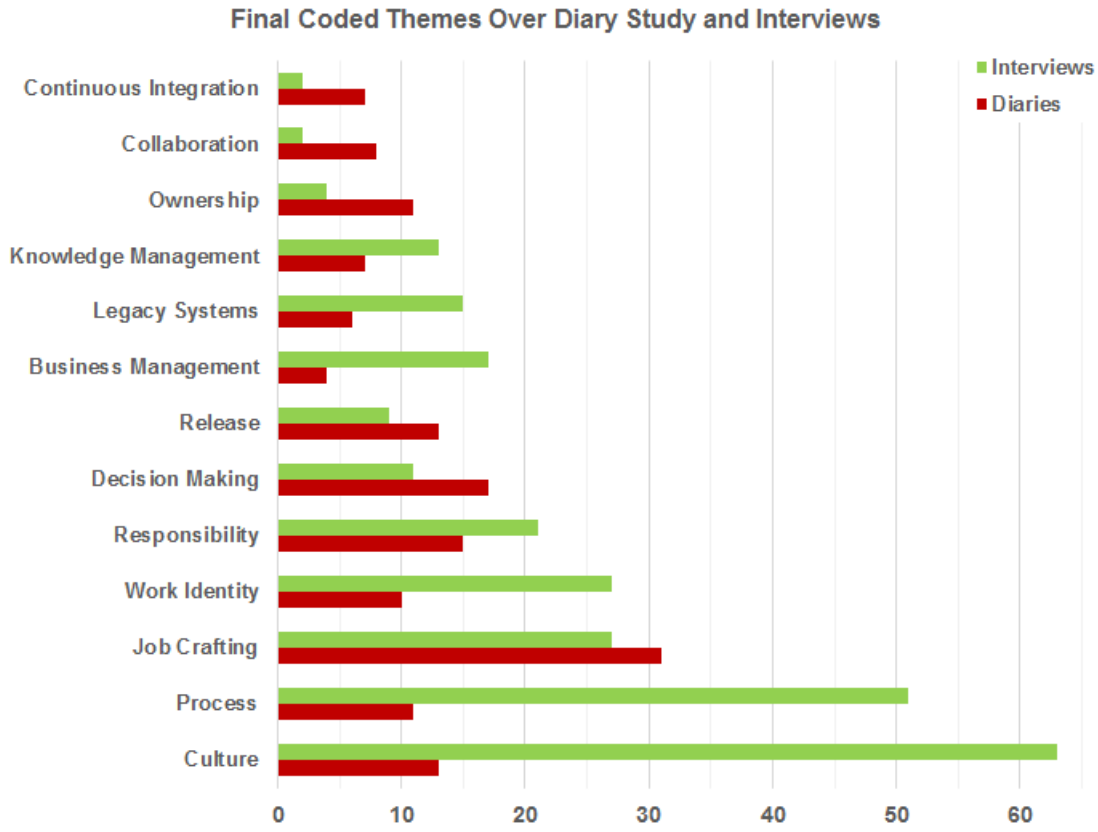


Figure 5.3: Frequency of final themes coded the from data collected at Anglia Farmers Ltd.

5.2 Case Study Time Period A

Covering the months of January to April 2016 inclusive, this section presents initial perceptions of DevOps, and what it means for AF. In addition, the maintenance activities on the legacy AFI system are explored, including the impacts this has on DevOps practice.

Cultural aspects of DevOps are also explored in this section, including the relationship between the Dev and Ops teams. Finally, some job crafting was observed within Dev, which is explored further.

5.2.1 Perceptions of DevOps

While the software developers employed at AF are familiar and comfortable working within a Scrum framework (see section 3.2.1), there is no consistent definition of DevOps within the team. While they had heard of the term, individual perceptions of DevOps were generally limited to three things: change, responsibility and deployment.

On the subject of change, participants state that they see DevOps as both influencing and directly affecting how they work. P10 states that it is "*blurring the lines between what constitutes development work and ongoing support, deployment and management of the real estate*" (I06). By 'real estate', P10 is referring to the physical IT infrastructure at AF. There is general consensus amongst the software developers that DevOps means both Dev and Ops would need to work more closely with each other, resulting in individual responsibilities becoming shared.

P10 also picks up on the idea of Dev and Ops integration. They state: "*responsibilities will merge and become everyone's responsibility*" (I06). P8 believes DevOps was all about the organisation cutting back: "*It's all to do with money and saving numbers... that's what I believe it is. If they can save money on support or programmers by doing something, they will*" (I46).

Senior managers at AF are more concerned with who is responsible for what in DevOps, although they had "*never heard of the term until recently.*" (I78). It is also stated that that DevOps has "*been driven more from our development team*" (I79), indicating that DevOps at AF is a Dev led endeavour.

5.2.2 Impact of Legacy Software Maintenance

As already set out, AF is reliant on a legacy software system, AFI, and has been for at least 10 years. Yet senior managers believe that AFI is end of life and no longer fit for purpose (I03). There was an attempt to update AFI and give it fresh life, but its quality rendered this very difficult to do. "*The lack of architecture*

meant you couldn't tease it apart and there were no layers... so you couldn't take this layer out and replace it, or I'll take this thing out and there would be a nice clean interface here that I could implement differently. So, it was a bit of a mess, and the decision was taken to re-write it" (I05). The decision to develop Harrier was therefore a result of the source code and architectural quality of AFI.

Yet due to its business-critical nature, AFI had to continue running during Harrier's development. This necessity also came with the ongoing requirement to undertake maintenance activities for AFI where necessary. Furthermore, an interface between AFI and Harrier needed to be developed in order to ensure data continuity.

However, this requirement to maintain AFI and the interface with Harrier introduced issues for both Dev and Ops. Firstly, AFI maintenance was generally reactive and was not formally included in Scrum sprints. Referring to sprints, P1 states "*I think AFI has been kept out of that. It seems (AFI), very... well I wouldn't even call it Waterfall, rather a 'do it as it comes' very reactive, I don't know what the word is for that to be honest. They're not doing it in an Agile way*" (I20). This also resulted in additional management issues, as the software developers disliked having to do any work on AFI.

P10 outlines the resentment amongst software developers with regards to AFI work. "*Certainly when certain people were working on AFI predominately, there was a bit of resentment ... like I'm not actually on the new project, and everyone else is getting to do this new, exciting stuff and they're stuck doing all this legacy Visual Basic (VB) code, which no one really likes*" (I02). Aside from the older technology being used for AFI, P10 once again raises the issues of working with bad quality source code and the difficulty of integrating this maintenance into the incremental approach taken for Harrier's development. "*The main problem with it is there is no separation of concerns... you can't pull one part out and replace it with another. You can't do incremental changes, so if you pull one part out... it's like tugging on threads, and it all starts to unravel*" (I02).

Frustrated with the disruption that AFI maintenance work causes them, P8 de-

scribes it like having to constantly change caps. "*Yes, as I'm learning the new technologies, I'm having to put myself into 'learning mode' and then, for example, when something has gone wrong with AFI or something hasn't gone right in testing, I have to then, take that cap off then try and get my head back into the other mode, and the swapping just takes a little bit of time. Obviously, when you're learning, things go out of your head and when you come back, you're like, well what was I actually trying to do and that's the hardest part, it really is when you're trying to learn. If I knew it all, it wouldn't be too bad, but learning it and swapping about is difficult*" (I45).

AFI maintenance is being undertaken in a reactive and traditional manner with some stark contrasts drawn with how development for Harrier is undertaken. P4 indicates, there was no agile approach with software development before Harrier. "*I remember when I first started it was sort of do this, deploy it and hope it works. Yeah, hope for the best! It really was like that.*" (I23).

Yet there was an appetite to improve the process and to ensure Harrier is developed in a much better manner. Commenting on the lack of Continuous Integration (CI) and lack of automated testing in AFI, P1 draws lessons for Harrier. "*The contrast is marked - no CI, few unit tests - and shows how important getting that build pipeline up and running really is. Thinking about environments and how to deploy code quickly to them is something that needs to happen right at the start of the development process*" (D19).

Such lessons may be valuable though as Harrier is a direct replacement, rendering it necessary to replicate functionality already present in AFI. However, given both the quality and technology differences with both systems, delays were introduced as complexity was overlooked. P3 reflects on this in their diary. "*Optimistically, I had hoped that this would take an extra week to deliver but there was far more to it than I had anticipated (i.e. there was far more going on in AFI that needed to be replicated on Harrier than I assumed) and it has ended up being an extra 3 weeks in total*" (D39). Moreover and crucially, P4 identified that the process of DevOps needs to thought through and automated testing, CI and deployment needs to happen from the start.

5.2.3 Goals of DevOps Adoption

The fundamental goals for adopting DevOps at AF is not only to develop Harrier, but to ensure it is of superior quality to AFI. P7 does suggest that just developing a better quality system doesn't make it DevOps, but being able to continuously deploy developed Harrier features to the business is attractive. "*A lot of it is not really DevOps, in that we're producing a much better system than we have currently, but the ability to deliver that system and keep it running is a big thing*" (I42).

In order to meet these goals, P7 does point to the necessity of shared responsibility from both Dev and Ops. "*I'm looking for automation down the pipeline, so I'm expecting the responsibility of the two teams will be to keep this automated pipeline running all the time with a fairly equal responsibility but obviously with an emphasis on Dev not to introduce crappy code that breaks it, and Ops to not fiddle with security settings without thinking it through*" (I41).

Additionally, and aside from sharing responsibility with Dev for the DevOps process, Ops are still required to provide more general IT support to the rest of the business, and as such are a crucial part of AF, as P6 has observed. "*A developer's never going to go and install a monitor for someone in the business, they (Ops) will always do that*" (I34). P9 provides insight that Dev actively collaborates with Ops to some extent with Harrier's development. "*We sometimes involve them at the starting point of any project for what would be the project requirements in terms of the technologies and hardware and everything*" (I49). However, P10 see Ops having a far greater role with releases. "*I don't see them being involved in the actual sprint which is developer focused. But I could see them being involved in the release*" (I10).

5.2.4 Change and Culture

Connections to Agile development approaches were also drawn as P6 not only talks about work flow, but also change. "*DevOps, how we develop as Operations*

I guess, how the Developers all work together, how we deploy, how we redeploy and stuff. It's all Agile, at least my take on it." (I30). They argue that "*anybody who's come from an old school approach to developing software might not embrace it initially*" (I32). The meaning of 'old school' is a reference to traditional approaches to software development, such as Waterfall (see section 3.2.1).

P3 acknowledges that "*AF has employed third party developers*" (I79) in the past, and Harrier is "*the first time we've done a big project with in house development and the team that we obviously have*" (I79). This brings to light the scale of change for AF, with it being "*a learning curve for senior management in the business*" (I79).

However, P10 indicates that DevOps related change is a scary prospect. Speaking about blurring the lines of responsibility, they state they "*always had comfort from the fact that there's a certain point you're not responsible for your work any more*" (I06). They continue in outlining that mistakes are opportunities for learning and therefore important for developers to embrace. "*If you don't live with your mistakes as a developer, you don't really improve as a developer*" (I06).

Yet, P1 also believes the process of any change will be slow at AF owing to the culture, and potential politics with a third party as well as managers across the business. "*The whole Azure thing, the whole third party who used to manage the servers. I think there's a lot of politics there too, that holds stuff up. It will be slow because there will be resistance from different managers and people who won't necessarily make the decision. Because they know the people in those companies, and you're much more likely to do business with a friend, than do it a new way. The 'I've been using him for 10 years' mentality*" (I19).

Despite this, change is evident according to P6. "*I can see that there is change at AF, definitely since I've been here... and how we work and how we get the business involved in every decision we make because it's going to save time and money*" (I33). Moreover, P7 sees Dev taking much more of a lead with traditional Ops work. "*The upshot is that Dev will lead all the deployment and configuration work except where Ops are needed to make changes that Dev do not have access*

to, e.g. DNS settings" (D12).

Such Dev led deployment and configuration work would mean utilising new tools, which P6 doesn't see as a bad thing. "*Just going to make my CV better aren't they, surely? Unless they build or get a robot to completely do my job and completely automate everything. I'm going to learn from it, and I think they need a tester. As good as Developers' code may be, there's always going to be integration and look and feel issues you know. So, it's only going to improve my skills.*" (I35). Here, DevOps is a potential opportunity for professional development. P6 also perceives that AF will benefit from DevOps induced change too. "*Business will be able to work quicker, they won't have system issues. They should be able to process more orders, things over the phone because the system will be better, they'll be able to get more work done in their working day, so it's certainly going to mean that we (AF) can take more business*" (I36).

Referred to as magic by P1, this process of automation occurred and worked first time. "*So the actual release procedure worked really well on the 24th. <name omitted> made a release bullet point list of about 12 things, <name omitted> handled anything data migration wise. I took the website offline, pressed my git flow button in source tree and the magic happened*" (D43).

5.3.3.1 Cultural Issues Between Dev and Ops

Prior to 2015, the software development team did not formally exist. Following AF's decision to bring software development in-house, there was one team which included software developers and systems administrators. However, both teams have been formally split, resulting in a "*view of 'ours and theirs' and 'theirs and ours'*" (I38). Additionally, P3 believes that a that a silo culture exists between Dev and Ops. "*There are still silos of Dev and Ops. I think... short of bringing someone in a DevOps role who bridges both parts, which potentially could cause more problems as you bring three people to the table. At least with two people you can kind of knock their heads together and agree that sits there and that sits there... which sometimes is what it's almost felt like*" (I80). Furthermore, a third

party provider still supplies physical infrastructure to AF, raising "*politics of how they fit in; what's their view of what we're trying to do; what their view is of working in the cloud*" (I38).

Generally, the Ops view of DevOps is positive. P12 states they "*would like to see it happen*" (I51). The view of DevOps is largely focused around communication and collaboration between both functions as they bring necessary yet very different skill sets. A further comment of "*there's not very many people who will actually do both*" (I51), illustrates a particularly niche overlap of any specific DevOps role.

Another key thing considered in the Ops view is that of physical co-location. The dev and ops teams are located within the same area of the building, with adjacent desks. They make the argument that cross-communication during normal working activities does occur which P12 believes can aid in any collaboration between the two teams. "*I might be talking to my colleague, they might be listening... or they might be talking and you hear what they're doing... and you go that's not going to work straight away.You can hear what they're actually saying and vice-versa*" (I53).

Another consideration is the understanding of each others roles, as P12 believes that misunderstanding of roles can form a barrier. "*My colleague, he's a bit more old school, so he might take an approach different to say <name omitted>, who has these new ideas. Or it could be that development do not fully understand what Operations is doing and vice-versa. We don't fully understand each other's roles yet and there has never been any full clear definition on who is responsible for or should take ownership of what bits*" (I65).

Ops at AF have taken a large degree of ownership and responsibility for the provision and support of software systems, including AFI and other Microsoft applications. P12 states that "*in house, we maintain it, we look after it, if anything goes wrong, it's our fault and we protect it and do anything with it*" (I67).

With Dev considering Microsoft Azure for hosting Harrier, P13 highlights a potential change for Ops as they have always looked after and supported both the hardware and software at AF. Furthermore, these systems have been hosted in-

house at AF with Ops having always been responsible for them. "*Development are very eager to get cloud bits and bobs going and they're saying we'll pay the money, we'll get Microsoft to sort it out for us <name omitted> has been looking after them for the past 5 years anyway, they're his baby, and now they want to throw them out of the window and go, we'll get a new baby. I think it's more about the protecting of his server and he wants to still be able to maintain it himself, than for us lot to sit there in the corners loose limbed and pay Microsoft. They are his pets, that's how I'd perceive it. A server is a server, once you start naming them, then you get sentimental*" (I67).

Yet, there are also perceptions that Ops are moving more towards supporting hardware. P6 comments "*I think IT support, maybe a year ago, would have been split Hardware / Software. But I think now, they're mainly moving, shifting towards the Hardware. When the issues get raised they run all the systems and do what they want. Obviously, they'll look at the ticket and then if they can fix it, they'll fix it. If they can't, they assign it to our team*" (I31).

Increasingly evident with DevOps is its inherent overlap with change and the necessity to manage it. P7 appears to have taken a lead in advocating DevOps adoption at AF. They are also "*pushing through the agenda and feeling ahead to see where it meets resistance and trying to then break that resistance down individually*" (I39). "*The agenda is to communicate the developer architecture vision to operations and what tools and processes are needed to make sure this works on Azure.*" (D05). In addition, P7 is seeking buy-in from senior managers with regards to the Azure platform for Harrier. They have been conducting analysis activities with different departments at AF. As such, they feel "*there is no outright opposition, it's more just inertia due to their own observations of a 'default position of sit tight because this has always worked, even though it's a bit messy'*" (I43).

In addition to seeking buy in from departments around AF, effort has been made to ensure communication regarding both Harrier and the development process occurs with departments either directly or indirectly through an internal development blog. P7 claims they are "*slowly winning people round to this new way*

of doing it" (I44).

5.2.5 Role of Senior Management in DevOps

In most organisations there is typically some degree of hierarchical structure employed, AF is no different. Figure 5.4 depicts the management hierarchy at AF within the boundary of this case study, showing that Dev report to the software development manager, who in turn reports to the head of group operations. However, Ops report directly to the same senior manager.

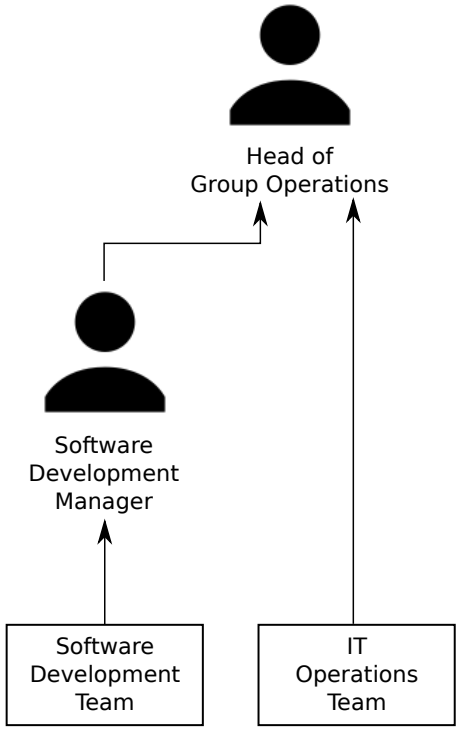


Figure 5.4: Observed reporting structure at AF within the case study boundary.

This hierarchy creates an issue as IT Operations can have as much influence on the DevOps process as the Software Development Manager, and have a higher reporting line than the software development team. Coupled with earlier acknowledgements that DevOps is developer led, a senior manager reflects that it is

like being between "*two main characters who don't always see eye to eye. I have to listen to <name omitted> from a support point of view and knowing there are some things he can setup that <name omitted> isn't 100% aware of. There are some things from <name omitted> from a development point of view that's in his language, and it's almost like I'm sat in the middle as a layman* (I81). In the same interview, they state that their approach is very much that of a layman and that they try to "*read it as this without trying to bring any technical terms to it*" (I81). The characters referred to are the software development manager and one systems administrator.

AF was also observed interacting with the local technology community too. In aiding the continuing professional development of the software developers, AF enabled and funded Dev to attend 'NorDevCon', a two day software development conference, hosted by Norfolk Developers[1]. While DevOps was a topic on the conference agenda, a presentation by a speaker from Aviva was of particular interest. P7 relates this talk with DevOps at AF, especially points the speaker made about DevOps needing commitment from senior managers if it is to be successfully adopted. "*He gave a really interesting talk, some of which really resonated with me regarding the situation at AF. Most interesting was his view that commitment from senior management is essential for the success of creating a DevOps working environment. Without that commitment, no matter how proactive the development team is, the barriers between Dev and Ops are not going to come down on their own.*" (D28).

The theme of senior management commitment is also picked up by P13, who believes that the organisational culture at AF needs to change if this is to be addressed. "*Our managers and the managers of them maybe don't respect or understand what we talk about all the time. That's another barrier above us, which can be a conflict. So that culture has to change there at some point, and it's not about if it has to change, but it has got to change*" (I69).

[1] https://www.norfolkdevelopers.com/

5.2.6 DevOps Driven Job Crafting

Senior management buy-in was continually sought in order to recruit a Business Analyst (BA) given both the size of AF and the development team. Pressure had been placed on P7 as they had to also undertake what analysis activities they could. "*With the size of the team we've got now, there is a place for a full time business analyst, and I've tried to argue that one. I've won the argument, but it's never transpired and hence one of the reasons I'm doing so much business analysis as it needs doing*" (I37).

The objective was more addressed to Azure Stack, a variant of Azure which AF can host itself, but had yet to be released. In relating to the senior managers, they were conscious to ensure communication was of a non-technical nature. "*After doing some reading up on Azure Stack (brings Azure cloud technology and benefits to on premises) I decided to run this past the senior developers, <name omitted> and the Ops team. I set up a meeting in a room with a TV and we watched a couple of Microsoft-produced videos on Azure Stack that were mainly non-technical in nature* " (D41).

The buy-in and involvement of Ops was one of P7's main objectives. With the consideration of Microsoft Azure as a platform for hosting Harrier, and attempts to involve Ops, P7 later states that "*Ops are not pro-actively looking to get involved in the Harrier roll out'*" (D17), producing a potential barrier to the DevOps adoption.

However, there is a strategic element to P7's dealings with Ops. In particular, P7's approach to achieving buy-in around the organisation (I44), harnesses <name omitted>'s desire to learn and acquire new skills. Indeed, they comment: "*Last week <name omitted> went on a three day PowerShell course. Mainly this was for him to become more productive in his current job. We have told him that these skills are transferable to Azure Power Shell and could be very useful in helping us automate much of our environments. Next time <name omitted>, <name omitted>, <name omitted> and myself are in we will talk this through and see if there is any interest in <name omitted> doing Azure PowerShell work*" (D25).

In using Azure, Dev have undertaken tasks typically associated with Ops. P1 notes that this is potentially harmful for promoting any collaboration with Ops. "*I do feel if Ops were more helpful on the Azure side we would be nearly a sprint ahead by now. I think the mistake we made was doing Azure ourselves. We are now seen as able to do it for now, but keep having to do more and more. A better approach would have been to have had early requirements supported by management on the Ops team. I think our technical intrigue as developers has actually hurt us here*" (D31). While evidence of Dev collectively job crafting through tasks, the reflection here emphasises the potentially negative consequences such actions can have.

The pattern of Dev undertaking perceived Ops tasks continues as P1 describes the tasks they've undertaken, which would have traditionally been performed by Ops. "*I made all the Azure web apps, Azure Power Shell Runbooks, added a config transform to each micro-service, added projects to Jenkins, Hipchat rooms, I think that's the big stuff. No Ops involvement was required*" (D38).

However, there are instances where issues addressed by developers have been communicated to Ops for their future benefit. One such example is concerning the provisioning of development environments on workstations and laptops. "*New laptop is here and has 16GB RAM, i7 and SSD. Seems a lot faster so far. What was interesting was <name omitted> got Ops to install a list of software. All was as expected except for SQL Server. We wanted Express with Management Studio, but got just Management Studio. So I fixed that myself and gave Ops the correct .exe to use for the rest of the teams' laptops*" (D54).

5.3 Time Period B

From May to December 2017, the relationship between Dev and Ops deteriorates. However, during this period a DevOps practice begins to emerge at AF.

The commitment of senior management becomes a prominent concern among the participants, alongside how the organisation, and in particular, the Development

team can cope with losing key individuals. This is down to two members of the development team giving notice of their departure from AF.

Meanwhile, maintenance work on AFI remains disruptive not only for Dev, but also Ops, despite this work reducing in volume. Instances of job crafting are increasing with both Dev and Ops. This is also explored alongside software developers believing their job roles are changing and their work identities transforming as a result.

5.3.1 Impact of Legacy Software Maintenance

Maintenance activity on AFI continues to be an ongoing necessity. A phased rollout of Harrier is evident, necessitating development of an interface referred to as the 'AFI RESTful Service' to ensure data consistency between the two systems. "*We have to communicate data from Harrier back to AFI, so there's that side of things and having to get it back the whole time. So, the single point of orders is within AFI. We call it the AFI RESTful service, and it sits here. It has some APIs that Harrier can hit, and it updates the AFI database*" (I77) (See figure 5.5).

Figure 5.5: The AFI RESTful service ensures data consistency between Harrier and AFI.

While necessary for data consistency across both AFI and Harrier, the AFI RESTful service does however add another software component to maintain. Despite this, the biggest perceived challenge appears to be the limitations with practice when moving between Harrier and AFI. As the development team moves forward with DevOps, P1 is concerned about the mindset connected to the manner in which the legacy system was developed. They are keen on "*getting everybody thinking in a much cleaner mindset, you know they're used to doing quite dirty hacks in AFI. So, getting them thinking about this is a clean project, we'll do it*

in a clean way... that kind of thing." (I74). This is also to preserve the quality and practice undertaken with Harrier development.

Yet, AFI maintenance work does appear to be reducing in volume, much to the delight of P9. "*I am now working more on Harrier than AFI. It has been particularly good to apply my skills with XML to Harrier too. The overall workload on AFI seems to have drastically reduced. In turn, I feel much happier to be working on Harrier than AFI now*" (D67). P8 also feels that AFI just wasn't worth spending much time on, citing its age, technology and quality as frustrating, placing limitations on what they can do. "*AFI is 10 years old, so it has its own things to do what you can do. Harrier has much more new things which you can do things in a much quicker and nicer way. It's just not worth spending as much time on AFI. It is frustrating, because you know you can do things better, but there is the case on quality and time periods*" (I90).

Ops on the other hand share many of Dev's frustrations with AFI. Difficulties for them include having to produce workarounds for end users, which are becoming increasingly common as Harrier continues to be rolled out in phases. P12 thinks about the workload for Ops, hoping Harrier will eliminate the need for these workarounds. "*There's ways round and <name omitted> has to find what users can't do. Harrier, I'm hoping, will eliminate that. So to a certain extent, my theory is that Harrier will make <name omitted> be able to do other things, rather than faffing around with AFI/Harrier. Call it Harrier, whatever you like. Harrier will hopefully replace the problems you have to deal with AFI. Then he would have time to do other things*" (I104).

5.3.2 Change and Culture

The relationship between Dev and Ops appears to have degenerated to a point where there is very limited communication between both. P12 notes that Harrier releases have "trickled" but comments: "*we've had nothing to actually support or a conversation or document to say this is what we've done, we're handing that over to you*" (I101). Dev are providing support at this stage for those currently

using Harrier, despite the P7's comments that Ops "*are now trained on Harrier and are starting to take support calls*" (D58).

Frustrated by the lack of communication and where Ops stands, P12 appears to be viewing this from the traditional AFI scenario, where developed software is passed to them. "*If you haven't handed it to us, how are we supposed to deal with it?! You carry on and support it, until you finish it or send us exactly what you want us to support*" (I101). "*Harrier is well from a build and needs to be done by development and then handed over*" (I102).

Commenting on communication, and regarding DevOps, P13 metaphorically describes the Dev-led aspect of it at AF. "*The only reason we hear about DevOps is through <name omitted>, but he doesn't manage Operations. I feel at the moment, <name omitted>'s got one size shoe that he wants DevOps to fit, and we're not Cinderella. I feel like in his mind, he knows what he wants for DevOps, but that might be different to how we see it at the moment. I feel we're not communicating enough to get any vision across. Although I wouldn't want to class us as ugly sisters... but yes. Regrettably, at the moment, I don't feel DevOps has moved as far forward as I would have liked it to*" (I105).

It is not all negative, as P13 respects the approach that has been taken, they feel there is bias towards Dev. Moreover, P13 believes senior management are not doing enough to moderate this. "*There's definitely been approaches towards it, but myself and <name omitted> are involved in development meetings. We've tried to involve them (Dev) in some of our bits as well, but it seems to be at the moment the idea of <name omitted>'s idea of DevOps to what we would like it to be is slightly different. Our manager isn't moderating that, so it's almost like a free for all. I think senior management and above, including the CEO... I think their role should not be just to moderate it, but to show by example. If they don't understand it or show interest, it will never motivate us to look at it*" (I105).

Conversely, P1 believes DevOps has led to an improved process with knock-on quality improvements of the software being written. "*The Harrier project is better because we embrace DevOps, and you know, we think about it as developers, and*

it makes our software a lot easier to write and you know, our releases have so far been a lot cleaner. So it's definitely improved things" (I73). Yet, they share the same concerns as Ops when it comes to senior management involvement. P1 also feels this may be down to senior managers not seeing the value of DevOps in the same manner as the Dev team does. "*I'm not sure that they're sold on the value of it in the same way that the developers are. I don't know why, but I think <name omitted> takes the lead on it really, rather than anybody above*" (I73).

Furthermore, it appears there is some perceived hostility too. P8 comments on an encounter they had with Ops when asking about server upgrades. "*The last time I tried to do anything on the Operations side, or put my nose in I got shouted at. I just asked Operations about some upgrades and if they've upgraded one of the servers to HTTPS, and I got moaned at saying its my responsibility, and then he goes, that for this I go to him, so what am I meant to do then?*" (I89).

Potentially compounding these issues is Ops having to move desks due to the space being needed by the Dev team due to the appointment of another software developer and a Business Analyst (BA). P13 believes that the desk move has worsened communication. "*I think there is now much less communication between Dev and Ops following <name omitted> and myself having to move desks due to lack of space, given the appointment of a BA and additional Software Developer*" (D64). Moreover, there is less involvement with senior management too and a feeling that P7 lacks interest in Ops. "*Our weekly meetings with <name omitted> have also ceased, and at present, have not resumed. <Name omitted> also stated to me that he does not feel it relevant for him to sit in meetings with us and <name omitted>, where we discuss Ops specific and facility tasks*" (D64). This has led to Ops feeling their input is not valued, nor do they feel in the loop with updates that would potentially impact them. "*While I agree that some development tasks are not relevant to us, I feel we do need to know if anything will affect the network, AFI or user experience*" (D64).

Although DevOps at AF may be Dev-led, P10 reflects that Ops should not simply be cast aside, despite the perception of them not wanting to be involved. "*On the one hand, if Ops have the attitude that we don't want to get involved, then it*

kind of makes it easier for us so long as senior management say well fine, they're not getting involved, then Dev can do what they want to do, and you can't object to it. It makes our life easier in some respects, as we get to pick and choose the things we want to do in terms of tools, techniques, processes and stuff" (I100).

The quality of Harrier doesn't appear to be affected, and is noted by Ops. In praising the quality and performance of Harrier so far, P12 appears ambivalent about Ops involvement. They state: "*with Harrier, Ops hasn't been involved too much, which I think has been a good and bad thing. I guess it's not working towards DevOps, but if there's nothing to fix, they are kind of doing it on their own stead. Realistically, to us, they are controlling it, and we don't have much input. To my eyes, it looks good, it performs well and from what I can see, the users are happy with it*" (I90).

However, P6 believes that Ops are such an integral part of AF that they simply cannot be excluded as they also offer substantial expertise and would be the first port of call for supporting Harrier's end users. "*But, I think as they're such an integral part of the company when it comes to fielding user queries and those kind of problems and things like that and the general day-to-day running of the office, they've got to be onboard, certainly with releases and what's going out. They need to know where to look through logs and things like that so they can relay better information to us. If they're going to be a first port of call to users coming in, if they know where the logs are, what the services are and what features are affected, they can say this things come in, here's the relevant log entries, just as a basic example*" (I100).

5.3.3 Role of Senior Management in DevOps

In time period A of this case study, a hierarchical structure (see figure 5.4) was reported. P13 reflects on this structure, believing it to be a problem. "*I think the reason it hasn't gone as well as we'd like it, is that both Dev and Ops should report to the same manager* (I105). In addition, a dominant view on DevOps from senior management revolved around the individual responsibilities. There

appears to be a mismatch of opinions, as P6, referring to the two separate teams, argues that senior managers should be the ones to define where responsibilities lie. "*Senior management need to specify the principle responsibilities of those different groups. It's all well and good saying its DevOps and it's combined, but there's two separate teams there who do things in different ways. So I think management's job is to specify where the dividing line is, even though with DevOps there's not supposed to be a dividing line*" (I93).

With time often very limited for senior managers, they are making key decisions which also include the infrastructure and tooling being used for Harrier. P6 states that senior management are "*broadly on board with the whole Azure platform, they're looking at doing this on-premises version which hasn't been released yet which Microsoft are looking to release later this year*" (I94). It is important to note that while Harrier will be deployed initially to Azure cloud, the decision from senior management places emphasis on the release and installation of Azure Stack within AF, thus adhering with the organisational culture of keeping as much in-house as possible.

The decision regarding Azure also sits very well with Ops. Despite having had no previous experience or exposure to Azure, P12 feels "*it's very similar to running a server, very much the same principle but you go about it in a different way*" (I103). They continue, believing this would be a good thing for AF too, while explicitly stating their preference to Azure Stack. "*I think it sounds a good thing. Because it's all lumped into one. It's one interface where you can do everything in one lump rather than fishing yourself around the server or creating the roles, where they are already there. And to me, looking at it, I would definitely have Azure running the same interface with us. But I'm more prone to having it internally than externally*" (I103).

P11 comments on how delays in these decisions result in delays to releasing Harrier. "*It is tricky to get time with <name omitted>, as they are also so busy supporting the business. <Name omitted> being our proxy, the key stakeholder in how AF is wanting the new system to be built does cause delay in readying work in time for Dev sprints. This of course will also cause significant delay*

to release" (D81). P6 echoes the concerns of P11 on delays in decisions having knock on impacts. "*We're waiting on stuff... we've waited for decisions and to have various sign off meetings. We can't progress further until things are signed off from higher up*" (I88).

The participants feel that senior management need to take a much more involved role within DevOps. Another issue is where responsibility for software development and the DevOps approach should sit within the AF organisational structure.

5.3.4 Key Personnel Loss

By the end of this time period, AF had to deal with an especially pressing issue. Up to now, the release process for Harrier has fallen largely on P1. This individual has announced they are leaving AF due to relocation and taking a new role. "*I've taken a job with Muddy Boots Software. They have 3 creaky codebases and want to bring in someone to oversee bringing DevOps and CI into their organisation, alongside a more micro-services type architecture. They've basically made a role for me, which is awesome*" (D88). Additionally, P6 is also leaving to embark on a freelance career. P10 reflects on this and the potential impact this will have on Harrier development. "*So we are losing two of our team - one to relocation, the other to contracting lucrativeness. The former is the real concern, as he has by and large owned a lot of the Azure related work within the team*" (D93).

While the departure of P1 and P6 is a situation perceived not to be uncommon in software development teams, the perceived skill loss concerns P10 the most. More importantly, they reflect on the ease and dangers of relying on one person for specific tasks and expertise. "*I guess this highlights how easy it is to rely on one person to get certain tasks done; when you are in full-flowing Dev mode, you don't stop to consider how certain things are getting achieved, just that they are getting achieved. So, this poses a bit of a problem for us in that we need to cover the impending skills loss. Ideally we will spread the responsibility across the team this time, and avoid a future repeat of this situation. But, alas, I fear this dance is performed in many development teams, over and over again*" (D94).

This aroused a sudden emphasis on capturing as much knowledge regarding the release procedures and other aspects that P1 was working on. "*My focus this last week had been about handing over as much knowledge as possible. To help with this <name omitted> allowed me to bring a handful of tickets into this and next sprint that I know will be particularly difficult or stretching in Ember terms. Other focus is on passing across some Azure and Jenkins experience*" (D95). It is evident from this that the Agile process itself was somewhat affected with specific tasks being drawn across two sprints. Furthermore, the knowledge management challenges with regards to DevOps were exacerbated due to the limited time before P1's departure. Invariably though, it leaves a void within the Dev team concerning Harrier releases.

5.3.5 DevOps Driven Job Crafting

While instances of job crafting activity appear to have increased, the strategy of to encourage greater involvement from P13 through using their desire to learn new skills appears to have backfired. While they attended a power shell training course, which AF had funded, P13 feels there was a misunderstanding and false expectation of what this would mean. "*I've been on a power shell training course. On the three day course I learned power shell. An interesting fact as I went to that to learn about active directory, exchange and group policy. But I think the impression for DevOps is that I'd be able to use that skill for Azure as well. So I think there was a bit of miscommunication there. I think <name omitted> expected me to come back and use power shell straight away for Azure. But the three day course didn't even touch on Azure. I've now got a book, with a big bit at the back of it, which is full of Azure*" (I108).

Perhaps this was a step too far as P13 reflects and relates this to previous experiences of programming when at college, which they disliked. Nevertheless, their desire to learn new skills remains. "*So, as I see more of this within Azure, it puts me off a bit where I see the simple couple of commands for active directory for reactivating an account or changing a list of active users or active computers*

running on the W32, then that's the thing I'm interested in. If it opens up in the future, I wouldn't mind delving into it. As a person, I've always wanted to learn more. But for my professional need, I feel I don't need that at the moment." (I108).

At this stage, job crafting appears be ubiquitous within Dev. Following dealing with an issue on the Azure platform, where a Microsoft software update affected the infrastructure used for Harrier development, P1 reflects on this and in particular, why they were the one to address the issues caused. "*I am guessing Microsoft updated the portal overnight and we didn't have something required in that version. Anyway the big question is why did I handle this? Its a virtual machine (VM). There are no scripts here and I was using a user interface with the Microsoft guy. Nothing about being a developer helped here. I fixed it because I want our nice front-end CI running again. We should reassert our push with higher management to be handing management of the Operating Systems (OS) and Azure to Ops*" (D60). Again, a sense of frustration can be interpreted from this refection, where it is felt that Ops should be handling issues such as these.

However, job crafting is evident with Ops, where they have taken responsibility for provisioning the 'Azure Reference' site, which is used for the testing and demonstration of Harrier by Dev. "*<Name omitted> and <name omitted> from the development team have been giving assistance to myself and <name omitted> on how to start and stop the Azure reference site. This is essentially the version of Harrier which is used for both testing and demonstration purposes. We can control this through commands, but in particular through the Hipchat tool used by the development team*" (D76). Until now, the relationship between Dev and Ops has been cold. Now, P13 comments about socialising well with the Dev team as well as joining them and others in team building days. "*Additionally, we are socialising well with the development team, and are looking forward to attending a crazy golf team building day with them and others in the business*" (D77).

The 'Harrier Implementation Group' was a suggestion to senior management from P7 to further involve them in Harrier's development and to coordinate activities such as user acceptance testing and training. "*I suggested to <name omitted>*

that a Harrier Implementation Group be set up to manage the roll-out of Harrier. We have our first meeting later this week. The group's responsibilities include User Acceptance Testing (UAT), training and platform - the last two of which normally are the responsibility of <name omitted> and <name omitted> respectively" (D90). A senior manager leads the group, and its members are made up from key individuals from across the organisation, including Dev and Ops.

Concerns have been raised regarding the limitations of AF's infrastructure, especially its internet connection. Further job crafting was exhibited by Dev as they explored this issue in collaboration with senior managers, including the CEO. However, P7 links back to the consequences of delays in decision making, as they argue a critical decision on the hosting of Harrier needs to be made. "*There is still no further progress on a hosting decision as <name omitted> has not yet arranged for InTouch to come in and talk through our options. My feeling is that the only sensible option would be on-premise while our Internet connection is anything but bullet-proof. Surprisingly, at my last meeting with him, the CEO seemed to be encouraging us to look at the cloud option - I think the 'serviced platform' idea is appealing*" (D92).

With the internet connection limitations becoming clearer, Dev and Ops appear to be settling on the direction of using Azure Stack. This potentially overlaps with the earlier observation from P12 indicating their preference was for internal hosting (I103). Perhaps Ops have been considering the limited internet connection from the start and aware of its limitations more than Dev or senior management had been previously?

5.3.6 Transformation of Work Identities

As DevOps practice continues to evolve within AF, developers are also beginning to see their roles differently. P10 comments that it feels odd to be just a software developer now. "*My focus in the intervening time has been almost exclusively on software development. The two largest features of AFI - ordering and invoicing - are in the process of being implemented in Harrier. The time I would have spent*

giving consideration to Ops issues, is instead being used to work with our BA. In a way this is a more traditional take on being a software developer, and it now feels a bit odd" (D69).

Another interesting observation was that of P10 who feels DevOps comes with a greater time demand on developers. "*<Name omitted> has largely taken over the Ops side of things, though I feel a sense of frustration at having to step away from a number of open issues. I think this highlights the extra demand on time that DevOps places on a developer. At present, I don't feel I can devote time to everything and still deliver on the development side of things*" (D70). This indicates a change occurring at a team level, where developers now, as observed by others, undertake perceived Ops tasks.

Reflecting on what their role entails, P4 comments on their achievements with the front-end of Harrier. "*Getting the Cascading Style Sheet (CSS) working on the front end was a big achievement for me too. I like to get involved in all aspects of Harrier, so the back end work is another string in my bow*" (D65). Having previously identified themselves as a front-end developer, P4 appears greatly motivated and self-actualised by the Harrier project. Now they appear to moving beyond the their original role which focused exclusively on front-end features by working on the back-end as well. "*On reflection though, I am definitely getting to learn, play with and apply new technologies as part of the overall delivery objective of Harrier. I still have a huge desire to continue learning too. Also, as I have generally always been a front-end developer this is new, given its back-end functionality, as such, I have been writing more C#*" (D82).

Furthermore, P4 appears to have developed a sense of ownership for the development of Harrier features. This is being driven primarily by their intellectual curiosity and desire to learn, investigate new technology and acquire additional skill. "*I've been working on an invoice pdf converter for Harrier. This essentially involves the conversion of Extensible Markup Language (XML) into a pdf invoice. Again, this is very new to me and the first time I've ever looked into such functionality. Nonetheless, it is great fun and has led me to investigating looking at FO.net(a C# library) as a possible avenue to developing a solution*" (D83).

While initially frustrated with the switching between both front-end and back-end development tasks, P8 also sees their role and subsequent identity at work differently. "*I am starting to find that the feature stories I work on are involving elements of both front and back end work. Subsequently, I no longer see myself as a front end developer, but rather a full stack developer, and I believe this makes me a much better developer*" (D84). Not only does P8 see this as a benefit to their own development, they find it very satisfying and of particular benefit to AF too. "*I also enjoy being able to move between both and I believe this benefits the business too, that operating in a full stack manner is more efficient. I also like the change too, if I did purely front-end for instance, I would probably end up getting bored*" (D85).

5.4 Time Period C

The final time period of this case study spans from December 2017 to the end of March 2018, when the research field work ended. DevOps at AF has become a Dev-led endeavour. While initially not involved, Ops have taken ownership and responsibility for providing support, not only to end-users, but also to Dev. While the relationship between Dev and Ops appears to have improved and become more collaborative, it becomes clear that the DevOps practice to emerge was coupled to Microsoft's Azure platform. This leaves DevOps at AF with an uncertain future following a senior management decision to cease using Azure.

Much to the relief of Dev and Ops, maintenance work on AFI has drastically reduced, albeit not gone entirely. Thus disruption caused by undertaking necessary work on legacy software has minimised. However, with the necessity of supporting two systems through a phased roll-out, the workload and pressure has increased for Ops.

Finally, job crafting is again explored across Dev and Ops. In particular, the BA appears to have catalysed the improving relationship between Dev and Ops. Additionally, transformation of work identities are explored, along with the influence

that job crafting has had on them.

5.4.1 Emergence of DevOps Practice at AF

P12 states Dev and Ops to be "*two separate departments, not working together as such*" (I134). However, they also acknowledge the Dev-led approach is working; "*at the moment it is working so we've actually got the basis so we know what's happening, we can actually get the updates from there which allows changing, we can configure to them, which we are getting*" (I249).

P10 also reflects on the Dev-led DevOps practice which has emerged at AF. "*It's ended up with the development team, taking on Ops' responsibility rather than Ops getting involved more in Dev, but we do include them in release notes, and things like that. We give them visibility of what's coming up, so they should know what's coming down the pipe for releases*" (I110). Supporting this view, P2 states that "*the software team are doing the releases generally with Harrier*" (I118). P11 goes further, highlighting the roles being undertaken by both. "*we've got a support team that should be supporting the floor and then we have our development team that is actually building, delivering and releasing the software out to the system, to the clients*" (I234).

P5 puts forward the notion that DevOps is all about software developers becoming self-sufficient. "*DevOps is where a team of developers become self-sufficient in terms of their IT operation* (I172). In the context of AF, they describe DevOps as "*a blending, a melding of the typical operations skills with the software development skills, certainly in Anglia Farmers, with a view to making us self-sufficient and more efficient* (I172).

P5 makes the argument that Dev's 'self-sufficiency' at AF means they "*look after their own destiny, they have their own capabilities to build, release, manage their environments, make their kit work and make sure they've got an environment that does what they need it to. I think a lot of this is about Dev taking on the Operations for their own environments*" (I122).

Supporting P5's argument on DevOps enabling self-sufficiency, P7 cites limited involvement from Ops, and much less from senior management. "*So I think it's been exclusively Developer-led, and the involvement of Operations has been fairly small. We've made an effort but it hasn't particularly been seized upon, and I think <name omitted>'s got other priorities so there was no real forcing of the issue, so it's just naturally flowed in a very Developer-led way*" (I127).

However, not all developers necessarily engage in the perceived DevOps practice. P1 describes DevOps as a multi-disciplinary practice that does not always appeal to all software developers. In these reflections, P1 also reveals a transformation in how they identify at work, referring to themself as a 'Devopeler', indicating their enjoyment of a role which involves both Dev and Ops related tasks. "*DevOps is the bit that some Devs like to do and some Devs don't. If people like to do it then they enjoy that grey line between operations and development, and enjoy setting up servers, scripts and all the kind of things that are somewhere in the middle. I'm a Devopeler, a Dev who does DevOps so sure, yeah. Whereas it's become apparent that some Devs don't want to do DevOps, and like just avoid it as much as they can at least from a Dev track.*" (I147).

As the study has progressed, there is generally wide agreement regarding DevOps at a cultural level. Communication and collaboration are critical, not just between Dev and Ops, but also senior managers.

Further progressing their feelings on DevOps, P5's comments support the multi-disciplinary view of DevOps that P1 has, alongside a view that cultural uniformity between Dev and Ops is crucial for increasing collaboration.. "*It means bringing together the two disciplines of Development and IT Ops, making them work closer together hopefully to get economies of scale, insight and cultural uniformity so there's more cooperation and collaboration*" (I120). This cultural uniformity extends beyond Dev and Ops though, as P10 believes senior managers can facilitate the culture and allow Dev and Ops to try things, knowing that sometimes it won't work. "*They have to give us the space to try it, the approaches that DevOps entails. They have to accept that sometimes we're gonna fail, because this is new to us*" (I113). On reflection, this credits the underlying culture of the organisa-

tion as it accommodated DevOps by affording sufficient freedom for Dev and Ops to explore, derive a process and shape their roles befitting of the organisation's goals.

P13, while citing communication improvements between Dev and Ops, also feels "*senior management should have played a bigger part*" (I143). While acknowledging that distinct Dev and Ops departments exist, P2 agrees that Dev and Ops are working well together. "*There are definitely still two very distinct departments, but yeah, I think we work well together*" (I154). Reflecting on the improved relationship between Dev and Ops, P7 places it down to a cultural shift with the focus on shared goals. "*It's the working together of people doing development tasks and operations tasks to keep the common goal of software, as it's being produced, being brought out into the production environments in a kind of way of working together*" (I193).

Describing it as joined up thinking, P3 believes AF now has a working DevOps practice. "*We've achieved it by I guess bringing two separate roles more closer in terms of the way that we've gone about the Harrier project. From earlier days they viewed life completely separately and I believe now they are much more joined up in thinking*" (I162). In referring to previous reflections on 'strong characters', P3 also believes progress has been made. "*I would say it's pretty fully joined up. It's thought through. I would use the word collaborative. There are still strong characters. I don't so much think that an intermediate is required. Whilst they're strong characters they've learned how to channel their views and actually both see the end goals*" (I170).

On the theme of joined up thinking, P13 also believes DevOps will "*build up communication between the Development team and Operations team, to collaborate on more projects and work*" (I261). This perspective of DevOps is shared by P12, who believes "*it's a collaboration between two different departments working together*" (I248). Despite P12's reservations over the previous desk move for Ops, located away from Dev, communication between Dev and Ops has improved with P13 reflecting on the desk move enabling an objective view. "*I think the communication between Development and Operations has improved slightly. In*

some ways Operations moving a little bit further away from Development has given us almost an out of the box view of it, and allowed discussion between both Operations and Development to be a little bit smoother" (I139).

P3 believes there has been a change in mindset, observing the changes evident with Dev and Ops. "*I think there's been a change in mindset, in working on an inclusive basis rather than an exclusive basis. I'm actually quite impressed how mature they've all been. I've not had to bang heads together, I've just had to sort of say, 'This will only work if you guys can make it work' and I think they've realised that themselves that 'it's going to cause me problems and if it causes me problems... Well actually if we just talk'*" (I163).

The involvement of Ops has also been reflected on by P7, suggesting Dev and Ops are working well together and no senior management arbitration has been necessary. "*It was just frustration down the line that we didn't really find a way of working together on anything other than first line support which I think, to be fair, we've now found a way*" (I202). The diaries and interviews suggest that the relationship between Dev and Ops has drastically improved out of the necessity to provide support for Harrier as it gets released to more departments at AF. P7 hints at ownership and responsibility play a big part in the solution. "*I think the problem then is it leads to, 'well we've tried this and we've tried this and we've tried this and it's all sort of, no we can't do it' so therefore we don't really try to engage particularly with things we've been told. We don't want to engage with that so I think we've found this nice balance at the moment with first line support, they're both quite happy with that. I think <name omitted> said explicitly that's where it starts and ends. So we both know where we are. So the frustration is now gone.*" (I202). Despite their physical separation, a shared goal exists with both Dev and Ops communicating, collaborating and supporting each other, and therefore delivering a holistic software development and support service to AF.

5.4.1.1 DevOps Practice Coupled to Tools

While evident that Anglia Farmers established a DevOps practice, it appears that Microsoft Azure underpins the self-sufficiency and cultural change of the Dev team. Ultimately though, the Azure cloud solution was abandoned as senior managers are more favourable towards Azure Stack, the self-hosted version. However, the ongoing delay in Azure Stack's release meant the decision was taken to cease using Azure altogether, in favour of Microsoft IIS, a self-hosted alternative. P1 perceived this may already be an upcoming problem. "*I think Azure Stack is going to be delayed until next summer, so it's not going to meet the time frames for phase 2 for us, so it's not an option any more*" (I152).

P7 laments on the senior management decision and what it means for DevOps with the perception of things going back to where they began. "*We then looked into Azure Stack which seemed that we could continue with the DevOps model that we had already developed (i.e. DevOps within the Development team with Support overseeing security). Unfortunately, this was not to be due to the delay in the Azure Stack roll out. Instead, some new hardware is on order to host Harrier internally. This more or less puts us back to where we were with AFI in terms of DevOps responsibilities between Development and Operations"* (D150). Moreover and crucially, the diaries and interviews reveal that the DevOps practice established has been moulded around the Azure platform, thus giving rise to P7's negative outlook.

Also reflecting on the consequences, P8 believes a backward step will be taken. "*If we have Azure the Developers were dealing with it, but as they're going back to in-house now, it's going to go back to Support, so technically they're in the same position as when they started*" (I210). Sceptically, P10 feels the decision is likely to be long term. "*I foresee us remaining on in-house hardware for a number of years, as the subsequent costs and effort of moving will always be weighed up in light of other business development needs (when you have a working platform, feature development will always take priority)*" (D115).

Claiming culture will once again become a major challenge in migrating to Mi-

crosoft IIS web server, P10 perceives two possible scenarios. One of these scenarios will build on the collaborative culture that has emerged, while the other is a reformation of silos. "*This will require Support and Development to work closely together to monitor and maintain the environment. Culturally, this would be the most challenging path to take, but may provoke the most change in how the two sides currently work. Or it could devolve into a living nightmare of passing the blame and fence building, but hey, best to be positive in our outlook*" (D116).

5.4.2 Impact of Legacy Software Maintenance

Although the decreasing frequency and low impact of AFI work has had a positive effect for Dev, the opposite is true for Ops. With Phase 1 of Harrier deployed to some departments at AF, P12 reflects on the impact of providing support on two systems. "*There's certain departments who are using Harrier and other people are still using AFI to do the exact same thing, which, I know that is moving over slowly but surely. You have a differential between working with AFI and then when you were explaining it, only to find out they're not actually using AFI, they're using Harrier*" (I137).

While support is being provided, the workload for Ops appears to have increased, with other areas potentially suffering as a result. Commenting on providing support for AFI, P13 likens it to firefighting and patching a sinking ship. "*I think AFI, because of how much firefighting you have to do, can take up quite a bit of time. You're patching a sinking ship when a new ship's being built, so you think, well, what's the point?*" (I146).

On the same topic, P12 cites AFI's dated technology as the main cause for support requests. "*If you didn't have to deal with AFI, in theory, it should be easier because we're going up the ladder with Harrier. AFI is out of date, so in theory you may not get as many problems with Harrier as you would with AFI. So if it wasn't there I would say that probably there would be less queries*" (I137).

5.4.3 Business Process Re-Engineering

While the amount of AFI maintenance activity saw a sharp decline during this time period, it remains necessary to maintain the legacy AFI system. Of the tasks being undertaken, P8 shares that these were minor and did not interrupt their working on Harrier. "*I did have to look at the RESTful web service for AFI in order to investigate why a few things were not working. In the end, there was an issue involving the wrong environment being used and issues around usernames and such. Thankfully, this was a relatively easy fix and did not interrupt my Harrier work. Otherwise, there has been no other AFI work*" (D98). Similar sentiments are shared by P4, which sits in agreement with P8's reflections. "*Sadly, I had to do a couple of AFI tasks, but thankfully these were small and did not interfere much with my Harrier work*" (D99).

While not interfering so much with Harrier work, the quality standards of AFI are far lower according to P4. "*When we develop something on AFI, as long as it works, it doesn't matter how it's done*" (I125). Conversely, and using AFI as an example, P8 believes that a high quality of source code is important in any environment. "*Code quality matters in any environment. If it becomes unmanageable or too complicated instead of taking five minutes to fix, it takes five days. So that's the situation with AFI, to actually do anything took you longer to undo the bugs that the change caused!*" (I214). Concerns regarding quality, mindsets and practice between AFI and Harrier were also previously reflected on by P1 in Time Period B of this case study.

P11 also identified major problems with regards to the business process within the invoicing department, which is similar to earlier reflections about how Ops have provided support to help users create workarounds to problems AFI could not offer a solution for. These issues included significant manual processing of invoices as AFI lacked functionality to process them initially. "*Well, I was shocked actually they were doing so many workarounds outside the system in order to get the information into the system, and that's what was shocking. But then, when you're trying to build Harrier to encompass all the rules, you can kind of feel why they're doing everything out of the system, because there's so many business rules*

based around that we're now having to program and actually put into Harrier, whereas in AFI it wasn't there at all." (I237).

P3 also echoed the clarity this has given senior management to AFI's limitations. "*It was built to process invoices that were correct, not to process every invoice whether it was correct or wrong and that's the subtle difference I guess*" (I166). AF's solution to these limitation was to increase staffing levels within the invoicing department, something which P3 also comments on, following deriving a more streamlined process as a result of DevOps, which has potential strategic impacts with substantial long term savings for AF. "*I envisage in a year's time that anybody who retires or decides to leave in the invoice office we won't be replacing and it will be a key driver for the business in terms of keeping costs of the operation down*" (I166).

While occasional AFI maintenance remains necessary, the intention is to switch AFI off. P3 says, "*Well we're still reliant on AFI at the moment. It still is, as far as I'm concerned, the point of truth. Obviously when we release phase two, Harrier becomes the point of truth*" (I171). Therefore the next release of Harrier will potentially be the pivotal point where it replaces AFI as the primary system in use at AF.

Nevertheless, and in considering continuity, P5 reflects on the limitations of external infrastructure. "*When the internet goes down at the minute, we would suffer. With the legacy system, it will still chug along*" (I190). While true in the case of Harrier hosted on a cloud based platform, the decision to host Harrier in-house was primarily down to issues concerning AF's limited internet connectivity. P10 also comments on the future of AFI, forseeing some necessary upcoming work, which may be the last on the legacy system. "*There's definitely a set of AFI work coming but it'll be a temporary thing and then at some point it'll be switched off or it'll just be left alone probably for historical reasons*" (I233).

5.4.4 Role of Senior Management in DevOps

There continue to be perceptions of little to no senior management involvement, yet they are by no means dormant. P11 explains that senior managers are making decisions with regards to how they want Harrier developed. But this can be frustrating for Dev, who would like to have more input into these decisions. "*They are making the decisions on what we're doing, how they want the system to be built and how they want the system to work and just specifying any additional rules that we don't know. It would be nice to have more input on the bigger areas of making the decisions, but yeah, senior management doesn't really worry too much about it*" (I236). These senior management decisions affect the nature of the system, its development and support.

Going further, P7 feels that DevOps is meaningless to senior managers and that they care just for Harrier being successfully developed and released. "*Outside of <name omitted>, DevOps means nothing to any senior manager. I think <name omitted>'s got a lot going on, which so long as software is being produced and released, I don't think that the efficiency of it is high up on their priorities*" (I129). P11 illustrates the potential consequences of this limited involvement. "*The person who holds all the information needs to be involved basically, but it does worry us because we're not able to move certain areas forward until we've got various answers and time, because the person who has all those answers has very tight time. It will delay the project completely and if we want to keep moving forward and try to hit some kind of deadline, then we need more involvement*" (I238).

It is evident within the diaries and interviews that there is a perceived detachment from DevOps by AF's senior management. While P13 feels both Dev and Ops understand the potential benefits DevOps can bring, they believe that senior managers do not. "*I think maybe both Development and Operations understand how much DevOps can help us but I don't think that senior managers or higher managers up there do. Obviously the business as a whole will probably have no idea what DevOps is as a grand scheme, because at the end of the day why would they? It only affects our two teams. But I think the lack of talking about it and*

for example ourselves, Development and <name omitted> speaking to <name omitted> about it. I don't think <name omitted> understands how much it could help us as a business. I don't think they fully understand what their responsibility could be with DevOps in that role" (I273).

While Dev and Ops appreciate that time is often limited for senior managers, they feel their involvement is often reactive. For instance, P11 believes there is some involvement from senior management, but this is often reactive following demos of Harrier development work. "*They're starting to engage, but I think it's from the demo. So as soon as they've seen what we actually have done, what we've actually produced, they've become more involved and want to see the end of it*" (I247). Illustrating a need for more pro-active involvement, P11 hopes for senior managers to become more involved and make key decisions therefore helping to avoid project delays. "*I'm hoping there's enough loud noises now being made by our manager to say, 'we need support', 'we need help', 'we need decisions and they need to get involved'. So now they're doing that. It's really late though. It needed to have been done at the point when we were jumping up and down back in December. We needed that involvement then. Now we've just delayed the project and unfortunately we also cut it in scope*" (I247).

5.4.4.1 The DevOps Champion

Progressing further from P13's feelings that senior managers should be more proactively involved comes the perception that DevOps was championed by the software development manager. While P13 exhibits a warm and positive attitude to the manner in which the software development manager pushes and promotes DevOps, they feel this would always be an inherently biased approach, irrespective of whether they were Dev or Ops. "*Because like any manager the first responsibility is to their own team so first of all they're always going to look to see how can they improve their team's efficiency and how it would benefit them. So I feel naturally that would always play bias towards whoever, even if the champion was in Ops' team, it'd be the complete opposite. The Ops champion would always favour their team, clearly.*" (I143).

However, P13 also looks favourably on the BA as a 'DevOps Champion' due to perceived impartiality inherent of their role. "*They might take a better approach to being a champion, because they seem to communicate better with myself and <name omitted>, rather than <name omitted>, and may have a more impartial take on both teams*" (I272).

Also apparent is the wider effect DevOps has on the organisation's culture, with P7 indicating the organisational culture is one of the biggest challenges to overcome. "*The main challenge is to get the key people in the business to sort of buy into this workflow at the right time and not to say, 'I haven't got time', 'I'm too busy' or need to sort of be there. So it's to get people outside the immediate DevOps type environment to buy into it working. I think they certainly like it when we do it. It's certainly not how it works currently*" (I206).

5.4.5 DevOps Driven Job Crafting

Additionally, job crafting has continued to occur within Dev, in particular where individuals are seeing beyond the departmental boundary of their roles. For instance, P4 now undertakes tasks outside of the original remit of their role "*I used to be predominantly working on front-end features and slowly moving on to back-end features*" (I186). In addition to task, P4 is engaging in relationship and cognitive job crafting as well given they express that the work they do is for the benefit of others. "*While an enjoyable undertaking, it was challenging too, as this was the first time I worked with the invoicing team, and mainly due to the differing terminology. The system they presently use is a bit chaotic, I believe Harrier will significantly improve things for them*" (D100). P8 has similar reflections although their focus on producing high quality work which makes other peoples' lives easier at work. "*I feel I am still improving my skills across the stack and am feeling positive about this. I maintain my focus on producing quality software and continue to be thorough in my approaches. On reflection, I like to think that what I produce makes others' lives easier. Additionally, when you see people using your software, and appreciate it, it feels good. This extends*

to other developers too, because good quality code is far easier to pick up (D113).

While identifying themselves as a developer still, P10 reflects on the broadening view of what it means to them. "*I'm still not a front end developer, I'm still not an expert at infrastructure, but I kind of take that view of all of it. I feel it's my responsibility to at least understand what the impact is at those stages and what the trade-offs are for accommodating those bits of the system*" (I117). This transformation of work identity is by no means isolated, evidenced by previous reflections from other developers regarding their roles. P4 also shares that their role at AF has seen them progress from working exclusively on the front end to working across the development stack. "*For me personally, it feels great that my development work column is currently clear and helping with others. On reflection, I feel I have come a long way since starting with AF; before I was strictly user interface (UI), but now like working with new technologies and working on different things. I also feel empowered to put forward my own ideas*" (D157).

Moreover P4 seeks the buy in of managers with regards to issues they perceive with the user interface of Harrier, indicating that they possess particular skill with this. "*One gripe I do have is that I wish the way the UI is coded, in that it needs improving. This is something I have spoken to <name omitted> about, and I feel I have skill with UIs. I think he is onboard with the idea. In particular, I feel we need to code the UI to cater for multiple screen sizes*" (D111).

From observations in this case study, the building of relationships was a pivotal component of the job crafting that occurred at AF. While the improved relationship between Dev and Ops is evident, there is an emotional element should such relationships end, as evident in P13's poignant reflection on recent Dev departures. "*On a more personal note, I was very sad to hear that both <name omitted> and <name omitted> were leaving. I valued the relationship and friendship I had with <name omitted> especially, and as a group we have had many social nights out which I'm sure everyone will miss. I look forward to attending the Developer leaver's meal*" (D103).

Job Crafting Benefits Being Felt

The job crafting at AF has led to benefits, which are being felt and reflected on. P9, who previously worked purely on backend functionality views working on the new technology both "*challenging*" and "*exciting*" (I133). Not only are individuals acquiring new knowledge and skills, AF are benefiting from more rounded skill sets and Dev and Ops employees gaining greater understanding of the organisation. P13 specifically mentions the benefit of gaining greater business insight into AF's operations and the role DevOps has played in this. "*I've learnt more business knowledge through understanding how Development work in their team. I understand DevOps is meant to help prevent conflict between two big teams like this and understanding and appreciating their views and concerns compared to our views and concerns, and seeing where there's a compromise with that*" (I142).

P5 also reflects on expanding beyond the original remits of their roles, and shows agreement with P13 about having a greater awareness. "*I get into a role and I start to expand out to areas where I feel competent. I'll certainly offer anything I've got and part of that just happens to be an awareness of how things are done elsewhere*" (I177).

With Harrier use increasing within AF, P2 shares their experience of interacting with users in the business with an objective of addressing issues experienced, something which never happened previously. "*As more and more people in the company are starting to use Harrier we're seeing more live bugs appear, a minimal amount, but there has been some. On two occasions I've interacted with people outside of my team to get more information about it to do some debugging at their PC where it's happening. I expect this to happen more and more, which is a good thing rather than hearing it second-hand not seeing it happen for yourself*" (D160).

Ultimately, any business can reap such benefits in terms of its workforce, but in the context of AF, DevOps may be the driver due to the organisational change that came with it. While being mainly based around exploration of tools and technology, P10 indicates that the underlying culture is an enabler, given the

freedom they were afforded to try things. "*We feel like we've got a bit more freedom to go and – 'We want to switch on this feature and have a look and see what it does'. Even though we're going to end up hosting in-house, we're still using the cloud for development and testing, which is great because that frees up a lot of bottlenecks in our development process*" (I222).

5.4.6 Change and Culture

While this case study reports a vastly improved relationship between Dev and Ops, a substantial degree of relationship building between the two departments was the result of the BA's ongoing involvement and communication with Ops. Despite not having a positive start, P11 has become the interface between Dev and Ops. "*They usually go through me if we need anything between Dev and Support*" (I245).

Seeking access to training documentation, P11 reflects on a moment of conflict with one member of Ops. "*Unfortunately a conversation with one member of Support was rudely interrupted by the other. I finished the conversation and walked off. Having previously worked in Support, the customer was always more important than current tasks at hand. Quite upset by this, however the Support member I was talking to did pop to my desk to complete the conversation*" (D114). Reflecting on their own experience working in a support role, P11 concludes with the view that it is critical for Ops to be approachable, and goes a long way. "*Approachability of Support is very important, developing good reputation and confidence with customers they are supporting can go a long way to making an efficient workplace*" (D114).

Nevertheless, this did not sway P11 from interacting with Ops. Following brief conflict with a member of Ops when querying a raised support ticket, P11 reveals an openness from Ops. "*Surprisingly the support member then comes back and provides further training to me on the file stream application. I suspect because I was pleased when I achieved what I needed to achieve on the initial request, it might have inspired them to want to help more. Later that day, the second*

member of support shares access to the Harrier training guide with this support member which is a surprise as they have not taken any interest in Harrier up until this point (D133).

Perceiving their role to also have become key in enabling DevOps to work at AF, P13 feels that P11 is a crucial communication interface between Dev and Ops. "*I think I've become by default, unofficially the communication bridge between Ops and Development, so I have quite a key role because that means I can communicate with Development in a very diplomatic way and also vice versa. If they have requests then obviously they can come to me and come to <name omitted>. So I think my role's been quite key in actually making sure DevOps works between the two areas*" (I262).

Further exploration of P13's perceptions point to a very positive and clear relationship development with the BA. Speaking warmly, P13 feels they communicate best with this person, and is very thankful to them. "*I think the person I find I communicate best with is <name omitted>, because they are very business-minded. I think both our mind-sets are very similar. I have a lot more knowledge of AF as a whole compared to them at the moment, because they're still relatively new, so they come to me for advice on AF, but they're really good at explaining what a good process is and how we can actually implement that. <Name omitted> is very good at just getting the point across straight away for myself and <name omitted> to understand*" (I263).

Sharing P13's view of improved communication, P12 also speaks favourably of the BA. "*I think the communication has come on a lot more. I want to see it improve because it's hard work getting information out if you don't actually have the conversation in the first place, but yes, it does, it's flowing. But I do think a lot of the flow is from <name omitted>*" (I257). Prior to the BA's arrival, much of the analysis workload was undertaken by P7. Acknowledging the additional workload this created, P12 appreciates the burden placed on P7. "*The role asked too much for what they were put in place to do in the first place*" (I260).

While DevOps can deliver working software rapidly, users remain the common

denominator. P11 feels that greater rapport, not just with users, but also Ops can only help with the uptake of released software. "*Encouragement of use of software has not been handled great, would have been good to see some floor walking as such of support to just help with issues as they happened. This would help to build a better rapport with users and Support*" (D124).

Reflecting on the overall Ops role, P12 comments on the necessity for Ops to support Dev and how they can only support users if they first support Dev with implementing the software. "*The only other way we work from the Development team is providing they have whatever they need to actually do the job. That's how I see it full stop. We support everybody within this building. But the aspects that they're actually doing are a small part, or if you like Harrier, is a small part of the whole business unit we support. But they need to give us the information to what they actually build so we can support it and support them to actually do it in the first place*" (I259). P12's reflections here demonstrates a substantial sense of ownership of the software support process at AF as well as hinting that there may no longer be silos of Dev and Ops.

DevOps Driven Transformation

At a team level, there is evident change within Dev as P10 states they have taken all responsibility for provisioning their environments. "*We've taken all responsibility for setting up our own integration environments, testing environments, like provisioning stuff in the cloud for that, and basically having more of an eye on how our whole system hangs together and how you can replicate all those parts somewhere else if we need another environment*" (I111).

This sits well with the earlier comments from P7 and P5 regarding the self-sufficiency of the Dev team. Extending to other roles, such as testing, P5 outlines software developers and testers collaborate in a DevOps environment too. "*I do have to build test data, test environments, but more I'm specifying to the Developers to help me build my environment. I'm certainly not a Developer with development skills. I have programmed in the past, I can do it, but they're quicker,*

better and I don't want to go and mess up something by building something not as good as they could do" (I123). Such collaboration is noted by P10, who has noticed a shift in individual attitudes, highlighting a greater sense of responsibility and shared ownership overall. "*I think people are more willing to get involved in kind of fixing problems wherever they happen to arise, so people aren't like, 'Ooh, I don't touch that bit of the system' or, 'I don't deal with the Azure bit'. Everyone kind of feels quite happy to take responsibility for various bits of it* (I115).

P3 views the improved skill and collaborative working of Dev in a very positive light too. "*Personally I think the skill set of the whole team has gone up and that's mainly through the feedback via call requests and this gives the opportunity to share some skills and knowledge and techniques between say new employees and existing ones who haven't been on the system before. So I think that's very, very positive"* (I212).

P2 also notes their role has changed as they have also picked work involving more collaboration with Ops. "*After learning a lot about the Ops side of the department from the handover with <name omitted>, I'm back on to primarily development, but handling the releases and candidate cutting as <name omitted> used to do. My role has changed so I that I can fill the role that <name omitted> left behind. I have more responsibility and am much more involved with Ops* (D108). P12 also has a new view on their role as they reflect on their now better understanding of DevOps at AF. "*I always understood my job role was, 'I'm Operations, they're Development, that's a clear-cut line'. Where now I understand what DevOps is, you kind of see how actually both are kind of intertwined together. And it obviously depends how much you deal with them, so I'm a bit more open minded than I used to be*" (I145).

5.5 Summary of the Case Study

Over a period of 14 months, AF's adoption of DevOps for the development of Harrier, the intended replacement system for AFI was studied. From the case

study alone, it is clear that DevOps is substantially more than just a technical phenomenon. Links to business management were rapidly discovered from the first time period, in addition to observed instances of job crafting amongst the software developers.

These themes continue in the second time period, where Dev and Ops are acknowledged to exist as organisational silos. Additionally, an already fractious relationship between both appeared to be deteriorating. However, perceptions of limited and reactive senior management involvement was a prominent concern to both Dev and Ops. This became critical especially as anxiety set in from losing two key individuals from Dev.

In the end though, these barriers were for the most part overcome, with a greater collaborative relationship between the Dev and Ops functions emerging. New strategies were implemented and a BA was employed, who played a crucial role in bringing both functions together as a cohesive whole. For instance, a shared responsibility for infrastructure requirements emerged with Ops working alongside Dev in beginning to automate much of the process of testing and deploying Harrier. As this collaboration grew, a single multi-disciplinary team emerged, thus breaking down the previously observed organisational silos. At the same time, collective and individual job crafting led to employees seeing transformations in their work identities, moving beyond the remits of what they were originally employed for. The organisation therefore benefited given the inherent mitigation of key person reliance. Furthermore, this job crafting was also actively encouraged by AF as the project progressed, realising the positive benefits it brought about.

While it became clear that an increasingly streamlined DevOps practice had emerged, it had a critical weakness being coupled to the workings of and focus on the Microsoft Azure platform. A senior management decision to cease using Microsoft Azure exposed this weakness and the inherent danger of coupling between process and tool. As a result, anxiety set in amongst Dev and Ops, leaving an uncertain future for DevOps at AF.

In conclusion, the AF case study reveals DevOps to be an interdisciplinary phe-

nomenon, and far greater than just the various tools used. While the DevOps practice that emerged at AF was primarily led by Dev, it was ultimately driven by business objectives, delivered by a multi-disciplinary team and fuelled by continuous feedback across both functions. Senior management involvement was a critical factor throughout the study and this needs to be pro-active if DevOps is to be successfully adopted. However, dependence on Microsoft Azure was a critical weakness of the DevOps practice which emerged. Therefore, this case study ultimately informs that an organisation should ensure process and tools are decoupled, and therefore not allow any tool to shape the way it does DevOps. Rather, any process should be system and tool agnostic so as to avoid the potentially catastrophic management and technical issues of allowing technology to determine any practice that emerges.

Chapter 6

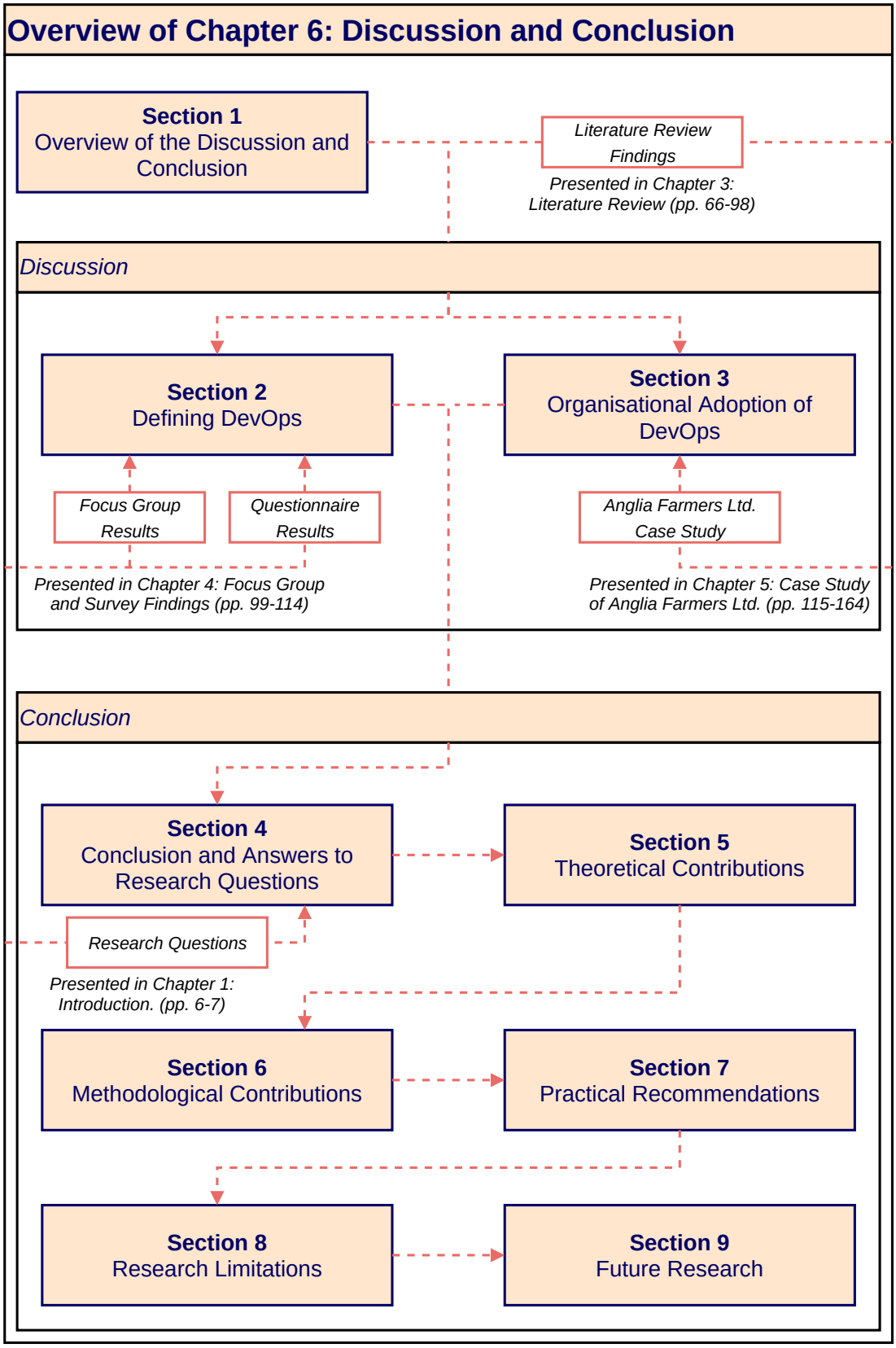
Discussion and Conclusion

"We can only see a short distance ahead, but we can see plenty there that needs to be done."

– Alan Turing

"The worthwhile problems are the ones you can really solve or help solve, the ones you can really contribute something to. No problem is too small or too trivial if we can really do something about it."

– Richard Feynman


Overview of Chapter 6: Discussion and Conclusion
Section 1
Overview of the Discussion and Conclusion
Literature Review Findings
Presented in Chapter 3: Literature Review (pp. 66-98)
Discussion
Section 2
Defining DevOps
Section 3
Organisational Adoption of DevOps
Focus Group Results
Questionnaire Results
Anglia Farmers Ltd. Case Study
Presented in Chapter 4: Focus Group and Survey Findings (pp. 99-114)
Presented in Chapter 5: Case Study of Anglia Farmers Ltd. (pp. 115-164)
Conclusion
Section 4
Conclusion and Answers to Research Questions
Section 5
Theoretical Contributions
Research Questions
Presented in Chapter 1: Introduction. (pp. 6-7)
Section 6
Methodological Contributions
Section 7
Practical Recommendations
Section 8
Research Limitations
Section 9
Future Research

6.1 Overview of the Discussion and Conclusion

This PhD research set out to explore what DevOps is, and how it is adopted by an organisation. The results of this research show DevOps to be interdisciplinary, primarily concerned with the rapid development and deployment of software, which it achieves through changes to the organisation, job roles and strategy. Therefore, DevOps has a substantial Business Management component.

In this chapter, the results from the focus group and questionnaire exploring the definition of DevOps (see chapter 4) are discussed and synthesised with findings from the literature. The discussion then moves onto the case study of DevOps adoption at Anglia Farmers Ltd. (AF) (see chapter 5). The discussion of the case study considers the findings on how DevOps is defined but more importantly, it considers the business management implications based on AF's experience and what it meant for the organisation's software development and support processes. This includes discussion on the organisational changes experienced by AF in addition to the observed transformations of work identities as a result of job crafting. Throughout the discussion, the results are synthesised with the literature reviewed in chapter 3, to move the discussion to a conclusion.

In the conclusion, the research questions posed earlier in the thesis are revisited and a summary of the key research findings is presented alongside the theoretical contributions this research makes to the DevOps knowledge. Additionally, methodological contributions for undertaking longitudinal empirical research with IT and software development practitioners are presented. Management recommendations are also offered in addition to acknowledging the research limitations. Finally, this thesis is brought to a close with the proposal for further Business Management and Software Engineering research avenues on DevOps.

6.2 Defining DevOps

Early research activities through the review of published literature suggested that DevOps is a difficult term to define universally. Roche [2013], claims the perspectives of DevOps are based upon one of two themes. Firstly, DevOps as a role with respective job descriptions and titles, for example DevOps Engineer (see appendix 13 on page 255). The other views DevOps to be an emerging concept that addresses the needs and demands of modern software development.

Roche [2013] argues that these themes are polarised, with one generally disagreeing with the other. Support is added as Ghezzi [2017] argue that DevOps practice is not mature, often informal and unstructured; while Fokaefs et al. [2017, 25:2] claim DevOps "eliminates the concept of a software life-cycle as a system undergoes changes with no interruptions to consumers".

It is argued that existing definitions for DevOps are unclear, ambiguous and sometimes contradictory [Dyck et al., 2015; Smeds et al., 2015], in turn further confusing what DevOps is and therefore exacerbate the challenge of successfully adopting it. Smeds et al. [2015] argue that the adoption of DevOps is not straightforward and the results from this research add support to Smeds et al.'s argument.

In this PhD research, the definition of DevOps was explored through a focus group and questionnaire survey with practitioners. Being able to define DevOps was a key and overarching theme necessary to explore it in context through the case study presented in this thesis.

Overall, focus group participants agreed that DevOps is a contextual phenomenon, meaning how it is realised for one organisation differs from another. However, a set of conceptual attributes that are core to DevOps, irrespective of context were agreed upon. Therefore, this research puts forward these attributes as a conceptual framework to help inform any definition of DevOps, taking into account the wider impacts and influences beyond the development, deployment and support of software. Participants also defined four distinct but non-hierarchical categories of which the conceptual attributes of DevOps were placed into. Crucially, they

agreed that DevOps must be driven by business goals but also continuously informed through a feedback loop (see figure 6.1). The value this framework brings to organisations is a guide to defining what DevOps means in their own context.

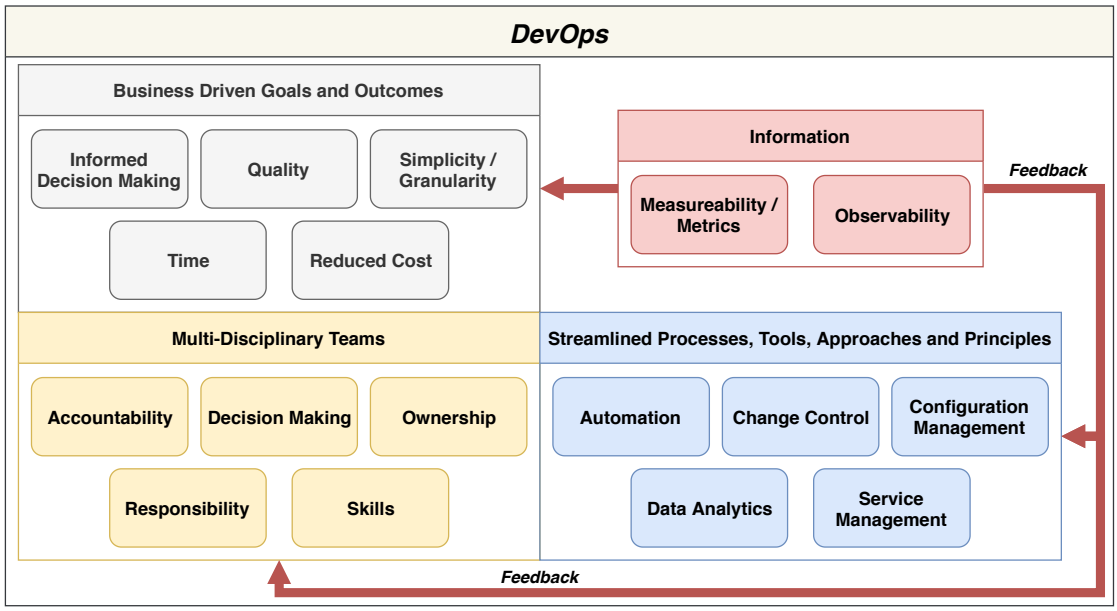


Figure 6.1: Illustration of the Focus Group's output of seventeen core conceptual attributes of DevOps.

Using the conceptual attributes they agreed, the focus group participants produced two definitions; one from scratch (see table 6.1) and the other derived following an evaluation of definitions discovered in the literature (see table 6.2).

Focus Group Participants' Definition One
"DevOps is a continuous improvement methodology that uses a set of tools, streamlined and automated processes, and empowered, multi-disciplinary teams to deliver, operate and inform business outcomes."

Table 6.1: Focus group participants' definition of DevOps created from scratch.

Definition one offers a more abstract view of DevOps which is based around the conceptual attribute categories. Conversely, definition two offers a more granular view focusing on DevOps as an evolution for the delivery of IT services, of which cross-functional collaboration between software developers and IT operations em-

Focus Group Participants' Definition Two
"DevOps is an evolution in how IT services are delivered and supported. It stresses cross functional collaboration to bridge the organisational process divide between development and operational teams. It aims to reduce the time between committing a change to a system and the change being placed into production."

Table 6.2: Focus group participants' definition of DevOps derived from Bass et al. [2015]; Dyck et al. [2015]; Mohamed [2015].

ployees is important. Finally, it provides the example of a software change being placed into a production system as quickly and reliably as possible.

Both definitions have overlapping themes as multi-disciplinary teams and cross-function collaboration can be argued as the same thing. Definition two's example of the rapid deployment of software changes can be linked to definition one's explicit mention of automation, as this would be the means in how this is accomplished. However, definition one explicitly mentions the business links and impacts DevOps has, which definition two only alludes to with the focus on IT service delivery.

The focus group output was further tested using a questionnaire which was sent to the wider DevOps community. While the questionnaire respondents showed no overall agreement with either definition, there was strong agreement with the conceptual attributes, especially Automation, Change Control, Observability, Reduced Cost, Service Management and Simplicity/Granularity. Taken together, the results of this portion of the PhD research therefore strengthens arguments that DevOps is difficult to universally define and its application is unique to the organisational contexts it is adopted within [Dyck et al., 2015; Smeds et al., 2015].

Smeds et al. [2015] put forward a number of capabilities and enablers of DevOps (see table 6.3). When taken together with the conceptual attributes produced by this research, Smeds et al.'s (2015) work is taken further, positioning how DevOps is an interdisciplinary topic and is not isolated to just the development, deployment and support of software. Rather, DevOps involves the wider organisation, being driven by its goals and culture. Therefore a continuous feedback

mechanism involving managers throughout is critical to DevOps.

Furthermore, the conceptual attributes alongside Smeds et al.'s (2015) capabilities and enablers, illustrate how DevOps can potentially contribute to building a culture of continuous service improvement. This would happen across the organisation through analysing metrics produced by software development activity in addition to continuous feedback [Dennehy and Conboy, 2017; Kim et al., 2016; Takimoto et al., 2016], but as seen within the AF case study, senior management commitment and involvement in DevOps is critical for this to happen.

Capabilities	Continuous Planning Collaborative and continuous development Continuous integration and testing Continuous release and deployment Continuous infrastructure monitoring and optimization Continuous user behaviour monitoring and feedback Service failure recovery without delay
Cultural Enablers	Shared goals, definition of success, incentives Shared ways of working, responsibility, collective ownership Shared values, respect and trust Constant effortless communication Continuous experimentation and learning
Technological Enablers	Build automation Test automation Deployment automation Monitoring automation Recovery automation Infrastructure automation Configuration management for code and infrastructure

Table 6.3: DevOps capabilities and enablers [Smeds et al., 2015, 171].

Tools associated with DevOps have received substantial research attention. This research illustrates while tools are an important consideration, they are just a small part of the bigger picture of DevOps and therefore neither advocates nor discourages any specific tool. Indeed, tools have a crucial role in the technical enabling of practice. But there is inherent danger of forming any reliance on specific tools or allowing them to influence and dictate any process, therefore coupling the process to tools. This is further discussed in section 6.3.1, specific to the case study at AF.

In short, the results from this research encourages any definition of DevOps to be made within the context of the organisation adopting it, while considering the conceptual attributes shown in figure 6.1.

6.3 Organisational Adoption of DevOps

In the world today software is ubiquitous and the needs of its users are often diverse and ever changing. The Agile approach to software development has been instrumental in making progress with addressing these issues, which were first encountered during the so called "software crisis" in the late 20th century [Randell, 1996, 70]. However, the emphasis has shifted with organisations motivated to not just deliver good quality software, but to do so in a rapid and continuous manner [Bass et al., 2015]. This fits well with the research from Pass and Ronen [2014, 80] and their argument of the "software value gap" which they define as the "unexploited potential for an IT division to increase value for the overall organisation".

6.3.1 Case Study of DevOps Adoption at Anglia Farmers

6.3.1.1 Why Did Anglia Farmers Adopt DevOps?

With software development activity for the legacy AFI software previously outsourced, senior managers at AF decided to bring the development of Harrier in house given concerns over the quality of the software. This decision was implemented by the appointment of a software development manager, who in turn recruited a team of developers with experience working within an agile software engineering environment.

The software development manager became a driving force behind adopting DevOps. The perceived performance benefit for the organisation was the rapid delivery of developed software which was of a good and usable quality. Senior

managers identified the strategic benefits and positive cost implications through analysing the metrics the DevOps approach was producing, further reinforcing the business management link, especially when it comes to decision making and planning [Fitzgerald and Stol, 2017].

6.3.1.2 Fluidity to Adjust Software Development Approach

Going beyond anticipated performance gains at AF, DevOps enabled the organisation to rapidly adapt its software development approaches in order to accommodate varying levels of complexity across departments within the business. For instance, the invoicing department was revealed to have many uncertainties, severe limitations with its use of the legacy system and no clear direction on its business processes. Yet, AF's DevOps process was able to readily facilitate a Kanban method for the analysis and development of invoicing features within Harrier. This flexibility of approach resulted in key requirements being identified and Harrier features being developed according to these. Furthermore, and more crucially with involvement from senior managers, DevOps enabled the business process for invoicing to be refined and streamlined, resulting in a significant reduction in wastage and identification of potential significant cost savings in the long term, thus informing strategic decision making.

Therefore, the AF case study offers support to the argument that such processes can promote operational excellence and greater strategic synergy [Sebastian et al., 2017]. DevOps at AF also included Continuous Deployment (CD) and Continuous Integration (CI), which while covered heavily in software engineering research, Chen [2017] and Sun et al. [2016] argue that both bring huge benefits to organisations too. AF realised these benefits through both a continuously improving development approach and being able to get software features into production rapidly and reliably.

6.3.1.3 Strategic Benefits

DevOps appeared to readily move beyond the locus of software development at AF, proving to be highly flexible while providing continuous feedback which enabled senior managers to identify issues in, and accordingly refine business process. In addition, analysis of the metrics produced by the DevOps process provided valuable insights and informed strategic decision making. Referring to the invoicing department, one senior manager shared their pleasure with the dramatic impacts DevOps has had in the refining of business process. Furthermore, the same manager projected that AF will not only recover the costs of recruiting an in house software development team, but will see a positive return. Ultimately, senior management attributes this result to the wider impacts DevOps has had, coupled with the ability to unlock and analyse a wealth of metrics, thus generating knowledge which has led to greater business performance and deploying better quality software.

The case study of AF reveals a positive return for the organisation from both bringing software development in house and the DevOps process which evolved for the development of Harrier. The invoicing department has severe limitations with the manner in which the legacy AFI system processed invoices; so AF increased staff levels to compensate. Thus this research supports the argument that "it can be difficult to fully automate deployment process due to context factors, such as the existence of legacy technologies" [Lwakatare et al., 2019, 228].

The methodological fluidity of DevOps allowed the developers to change their working method specifically for the invoicing department to allow for in depth business analysis and requirements gathering. This in turn resulted in senior managers seeing first hand the inefficiencies of the invoicing department and the limitations of AFI. Subsequently, the invoicing business process was streamlined with Harrier developed specifically to meet the needs identified. This links well and supports the argument from Pass and Ronen [2014], that IT has a potential to increase value for the overall organisation. In AF's case, this was strategic value with the facilitation of streamlining business process and the potential realisation of substantial long term savings. Furthermore, this highlights the value that a

Business Analyst (BA) can bring to software development functions as well as organisations.

The AF case study also highlights the inherent relationship that DevOps has to the discipline of business management, in that DevOps should have a strategic alignment with the organisation [Dennehy and Conboy, 2017; Fitzgerald and Stol, 2017; Fokaefs et al., 2017]. This also fits well with the previous section about how DevOps is defined, and that business goals should drive and be informed by any DevOps practice. Taken together, this research adds strength to the argument that DevOps is an interdisciplinary topic in both application and research. Moreover, the alignment with business should also be continually assessed and improved, through continuous learning, as put forward by Fitzgerald and Stol [2017, 176].

Synthesised with Takimoto et al.'s [2016] model, the results of this research proposes an abstract model of DevOps, taking into account how organisational decision making, goals and strategy are a major driver (see figure 6.2).

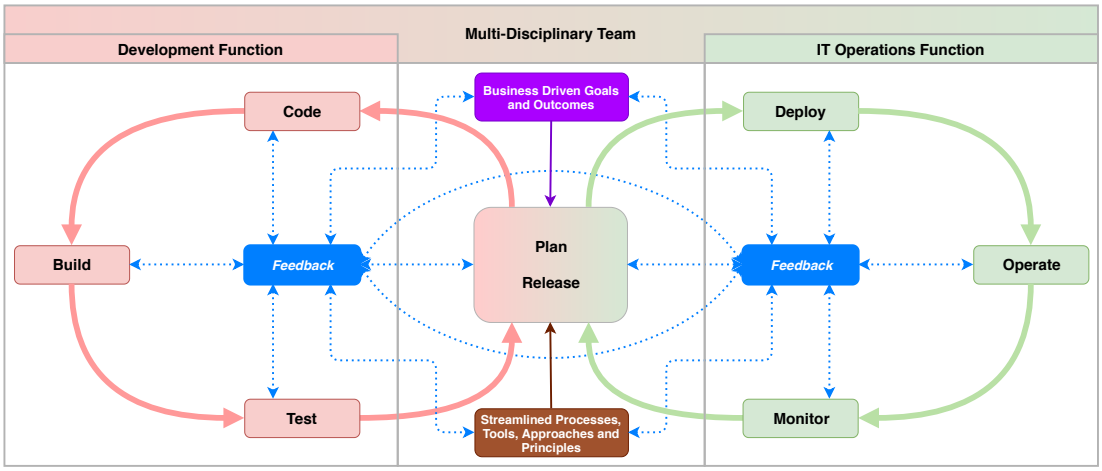


Figure 6.2: Iterative process of DevOps and harmonisation model derived from the results of this research and Takimoto et al. [2016].

While the adoption of DevOps was not AF's primary driver for the development of Harrier, the organisation did experience benefits such as good levels of automation arising from a combination of the Microservices architecture and DevOps practice with Harrier's development. While Koilada's (2019) case study organisation

innovated as a result of DevOps, the AF case study argues that commercial innovation is context dependent. Nevertheless, it does indicate DevOps is a potential process innovation, thus alluding to further study in this area.

This research postulates the view of software development and IT operations being represented as functions of a multi-disciplinary team or unit, rather than specific and separate departments or teams [Veres et al., 2019]. So for instance, individuals with roles in these functions which can include: business analysts, infrastructure engineers, managers, software developers, systems administrators and testers work collaboratively as a cohesive whole, taking a shared ownership and responsibility for meeting an agreed common set of goals that have a strategic alignment with the organisation [Dennehy and Conboy, 2017; Fitzgerald and Stol, 2017; Fokaefs et al., 2017]. While this collaborative mindset was argued as necessary by McLarnon et al. [2014], the focus was limited to shared goals for only software developers and systems administrators.

The relationship between software development and IT operations is a common theme in the literature, and what Kneuper [2017, 79] describes as a relationship with "notoriously difficult interaction". This research supports arguments that the relationship between software development and IT operations functions can be difficult. Furthermore, the findings from this research argue that senior management should also be considered in the relationship between software development and IT operations, given this appeared critical for AF. In short, and from a socio-cultural and socio-technical perspective, issues around the relationship between these business functions need to be tackled, especially in a medium sized organisation such as AF, in order for DevOps to be adopted successfully.

While this research argues that some understanding of each stage in the DevOps process depicted in figure 6.2 is necessary, individuals would still specialise at specific stages; but they would also be able to assist elsewhere if necessary. For example, a systems administrator may specialise in more traditional IT operations tasks such as infrastructure, deployment and operation of developed software with the provision of end user support. Within DevOps, they may also work more closely with software developers in order to write small applications or scripts

to automate the provision of, and deployment of developed software to run-time environments. As observed at AF, planning and release involved the software developers, systems administrators and a senior manager, who communicated the business goals and desired outcomes. In turn, this stage is inherently informed by feedback from all other stages in addition to the processing and evaluation of various metrics collected during previous iterations. The information these analyses provided enabled DevOps to evolve in order to accommodate the business objectives and requirements of AF.

Figure 6.2 also shows the nature of this feedback as a continuous loop working alongside each stage of the DevOps model. Therefore, continuous feedback as a "capability and enabler" of DevOps [Smeds et al., 2015, 171] is too limited. Instead, this research argues that continuous feedback is both a central and critical component of any DevOps implementation. Further support for this argument can be drawn from the discussion in section 6.2, where focus group participants explored how DevOps is defined and implemented.

Likewise, a software developer could engage directly with end users when they encounter issues with deployed software, thus providing support which could traditionally be provided by a systems administrator. In the case of AF, this was observed whereby a systems administrator could not offer a solution to issues being experienced by users of Harrier. Rather than simply pass off the problem to software developers, as is typically the case where "most IT services are organised in silos" [Airaj, 2017, 2], a business analyst and software developer became directly involved and worked collaboratively with the systems administrator to observe the issue in context. While such observed collaboration is by no means exclusive to DevOps, it does add support that it is linked to core business system transformation and the utilisation of Agile process, technical capabilities, expertise and skills [Karl et al., 2016; Sill, 2015; Smeds et al., 2015; Takimoto et al., 2016]. Moreover, a collective ownership and shared responsibility for problems was exhibited, which Smeds et al. [2015, 171] argue is a "cultural enabler" of DevOps. The result of this observed collaboration led to problems being identified, a solution implemented and deployed in the next release of Harrier.

The continuous nature of DevOps across functions means any development work on new features or solutions to experienced problems are not only rapidly implemented, built and tested, but also continuously released and deployed to users. The very nature of the continuous delivery of software can enable the provision of timely and effective support from the IT operations function, therefore adding value through "continuous service improvement (CSI)" [Takimoto et al., 2016, 9]. Collectively, DevOps has a strategic component for organisations, highlighting the value IT can potentially add to the organisation, thus addressing the so-called 'software value gap' [Pass and Ronen, 2014, 80].

While AF adopted DevOps for internal software development activities, the results and literature offer support for the argument that approaches such as DevOps could indeed become "a competitive necessity" [Sebastian et al., 2017, 205], thus shifting the focus to not just delivering good quality software, but being able to delivery it rapidly and reliably too. This concurs with the argument from Bass et al. [2015], claiming that competition is as a major motivation for organisations to adopt DevOps.

While the conceptual and abstract view of DevOps was generally well received at AF [Jones et al., 2016], the results of this research show that adopting DevOps is not straightforward and is a long-term activity, especially for a medium or large organisation [Lwakatare et al., 2019]. AF quickly realised the substantial change, cultural and job role implications which DevOps introduced, especially for software developers and IT operations functions, which section 6.3.2 discusses further.

Another key finding to emerge was that pro-active engagement from senior managers was critical to the successful adoption of DevOps. There were times at AF where participants felt senior management could have had much more input, especially where there were disagreements and resistance. Without the business input and conflict resolution such individuals in the organisation can provide, the level of DevOps induced change could potentially lead to projects failing and intensify existing organisational silos [Airaj, 2017]. Moreover, the results add support to the argument that DevOps has a strategic component [Sebastian et al.,

2017] and that it should not just be about developing software and nothing else [Karl et al., 2016].

6.3.1.4 Danger of Tool/Process Coupling

In software engineering, coupling means how interdependent software constituents are; in other words, how close one part of the software is connected to another. High levels of coupling means more effort is required for maintenance activity which also carries greater risk. Highly coupled software was one of the major factors of the "software crisis" in the late 20th century [Randell, 1996, 70]. However, it goes further as culture and practice can be formed around the manner in which software is developed, including a reliance on specific tools. This forms a high degree of coupling between tools and process, which can jeopardize any software development project and be increasingly difficult and risky to alleviate [Airaj, 2017; Chen, 2017; McLarnon et al., 2014; Roche, 2013; Sebastian et al., 2017].

Previously in this thesis, a collection of conceptual attributes for DevOps were identified and discussed (see section 6.2). These attributes were evident at AF, especially through the increased automation of software testing and deployment, greater control of change as a result of software development activity and the eventual realisation of reduced operating costs. Furthermore, the microservices architecture which made up the Harrier system offered greater simplicity and granularity to software development and maintenance activity. While Microsoft Azure played an important part in enabling greater levels of automation with software deployments, it was also a critical weakness of DevOps at AF. This resulted from a cultural difference between the software developers, IT operations and senior management. While used to great effect by the software developers, Azure played a big influence on how DevOps evolved at AF, thus becoming highly coupled.

A management decision to cease using Azure became the biggest threat to the tool-coupled DevOps practice which had emerged at AF. At the end of the re-

search fieldwork, DevOps practice was disrupted, leaving it with an uncertain future and delays to Harrier's roll-out due to the entire process having evolved from the manner developers worked with Azure.

Nevertheless, this research does not dismiss the role tools play, indeed they play a crucial role in realising DevOps. Ultimately though this research strongly argues that tools must not be the locus of DevOps. Rather, DevOps should be a conceptual approach and tool agnostic. Put simply, this means that any DevOps process should not be determined by the tools being used and it should be possible to change tools with minimal disruption.

6.3.2 DevOps Driven Job Crafting

Earlier in this thesis, job crafting was introduced as a theory in business management defined as "the physical and cognitive changes individuals make in the task or relational boundaries of their work" [Wrzesniewski and Dutton, 2001, 179].

Task crafting occurs when an employee makes changes to their job's task boundaries, namely the scope, type or quantity of job tasks employees choose to do. Relationship crafting occurs when an employee makes changes to who they interact with in the workplace and the quality of any interaction. Finally, cognitive crafting occurs when the boundaries of their job are changed by an employee, altering how they "parse the job" in the sense that they either view it as a "set of discrete work tasks or as an integrated whole" [Wrzesniewski and Dutton, 2001, 185] .

The concept of "Work Identity" refers to how employees define themselves in the workplace, for which Wrzesniewski and Dutton [2001, 186] argue job crafting has the potential to shape. Because of the actions employees take to redefine and shape their jobs it can therefore change the way they define themselves in the workplace. Furthermore, job crafting is argued to be socially embedded [Berg et al., 2010] and is not necessarily exclusive to individuals, but may occur at a team or organisational level too [Mäkikangas et al., 2017]. Changes to how they

perceive their job can radically and fundamentally change how an employee approaches it and can be linked to motivations around self-efficacy and performance, as was identified by Tims et al. [2014].

The term 'stack' is used to describe how various pieces of software which make up a system interact. Traditionally, software development roles focus on one area of the stack such as back-end or front-end. However, software developers in particular are regularly faced with technological change and innovation [Chilton et al., 2005]. Full stack developers go beyond this traditional remit, focusing development activity across all parts of the stack. Software developers at AF were originally employed to work on either back-end or front-end software development.

AF's adoption of DevOps, together with senior management encouragement to explore their roles, appeared to drive job crafting for the software developers. This was observed in the first instance as task crafting, where software developers took a holistic approach and responsibility to develop features across the stack, thus reshaping the boundary of their job and becoming movers of the Harrier project [Wrzesniewski and Dutton, 2001]. Additionally, developers began to see their work not just about delivering output as per their original job descriptions. Instead, they began to perceive their work identity as Full Stack Developers, where their work focuses on both front-end and back-end software development. This also fits with cognitive job crafting as the software developers assumed a holistic responsibility and ownership for the development of entire features for Harrier.

Relationship crafting was observed as software developers perceived their work on Harrier was a vital part of the organisation [Wrzesniewski and Dutton, 2001], and therefore began to collaborate more with end users, IT operations and senior management. Likewise, the same was true for IT operations and one senior manager, especially when it came to collaborating for development decisions and addressing complex end-user support queries.

As job crafting continued at AF, one software developer's work identity transformed a second time when they considered themself to be a 'Devopeler', making a

specific reference to this meaning they are a software developer that engages with DevOps practice. This employee was observed to have greater perception of their job having a greater collaborative involvement with IT operations in addition to performing tasks, such as automating infrastructure provisioning, continuous integration and software deployment. Cognitive job crafting was apparent too as this individual saw their job role as being much more than that of just writing software, but rather taking more ownership and responsibility for the release, deployment and support of the software as well [Wrzesniewski and Dutton, 2001] (see figure 6.3).

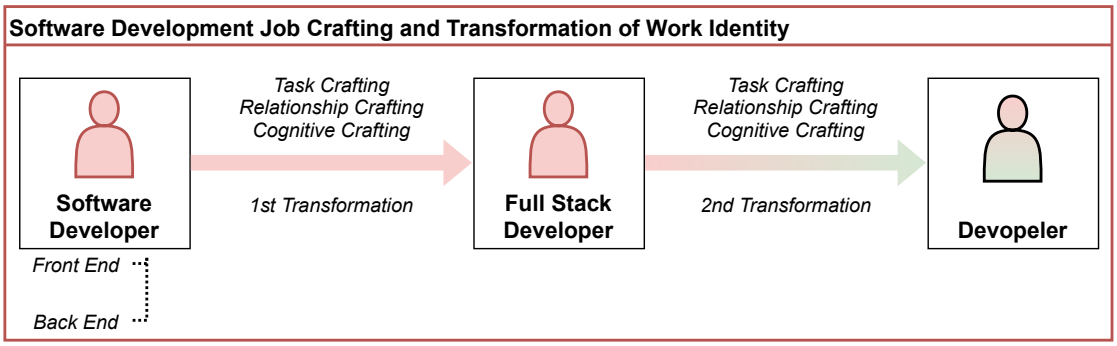


Figure 6.3: DevOps driven job crafting and work identity transformation proposition for software developers.

Based on the case study of DevOps at AF, a 'Devopeler' is a natural evolution in work identity from a full stack developer. Therefore, this thesis puts forward that the perceived job role as a 'Devopeler', while inherently cross-functional and full stack, includes tasks and relationships outside of the traditional remit for software developers.

No distinct transformation of work identity was observed for the systems administrators. However, following the job crafting observed at AF, this thesis argues that systems administrators have the same potential to job craft and thus form a new work identity as a result of DevOps. One systems administrator, engaged in task and relationship job crafting, moved beyond the traditional remit of deployment and support, thus working in increasing collaboration with software developers to automate the provision of IT infrastructure and deployment of software.

As the AF case study progressed, the systems administrators were undertaking tasks aligning with the duties expected of a DevOps Engineer (see appendix 13 on page 255), which is a job role many organisations have been observed recruiting for in the last decade. While no work identity transformation was observed within IT operations, this research proposes that job crafting would not simply cease, but occur along similar lines to software developers, potentially leading to a further work identity change to a 'Devopeler' or something similar whereby they see their role having greater collaboration and cross-functional working with software developers (see figure 6.4).

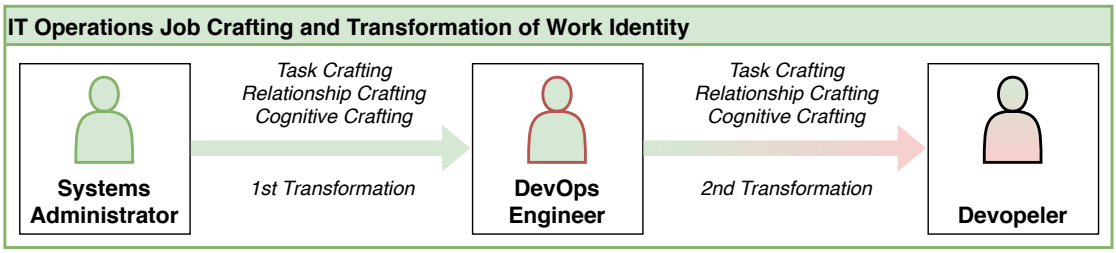


Figure 6.4: DevOps driven job crafting and work identity transformation proposition for IT operations.

While appendix 13 offers an example of what is required in a DevOps Engineer role, there appears to be little distinction and precision on what the role and title means, despite it gaining more ubiquity in recent years. Therefore this research postulates that the contextual definitions of DevOps also applies to any associated job roles within the organisation's context. However, arising from the AF case study, the term 'Devopeler' is new and implies an integration of software development and IT functions.

The observed task and relationship crafting at AF challenges the common perception of what the IT Operations function is and means, therefore offering strong evidence that DevOps intrinsically tackles issues around organisational silos [Airaj, 2017; Kneuper, 2017]. Ultimately though, job crafting was a positive side effect for AF's adoption of DevOps, which was driven by overarching organisational goals, greater emphasis on collaboration and shared responsibility for the rapid development, deployment and support of software.

This research strongly argues that continuous deployment is a cornerstone constituent of any DevOps implementation. Claps et al.'s(2015) model highlights a number of technical and social adoption challenges of continuous deployment, which include changing team roles and coordination, thus are inherent to DevOps. The case study of DevOps at AF was the first to apply job crafting in a software engineering environment where DevOps has been adopted. Moreover, job crafting serves as a good theoretical lens to observe in particular the business management implications of DevOps. Furthermore, the AF case study shows the job crafting which occurred was socially embedded in that it also occurred at a team level [Berg et al., 2010], and was not in any way inhibited by managers.

6.3.3 Theoretical Implications for Job Crafting

These findings have several implications for job crafting. As a theoretical lens, job crafting aided in explaining the nature and motives for the changes software developers at AF made to their roles, and the resulting transformations to their work identities.

Wrzesniewski and Dutton [2001] maintain a neutral position as to whether job crafting is good or bad for organisations. The case study of DevOps adoption at AF supports this position as job crafting was not suppressed by managers and therefore affording software development and IT operations employees the freedom to explore what DevOps meant for their roles as well as the organisation. Aside from being able to acquire new skills and do things they would have not done under the original remit of their roles, employees strongly felt that senior managers should not be completely hands off, especially where key decisions need to be made.

Overall, the job crafting observed in this research while not negative, does confirm that it can lead to transformation of an employees' work identity and provide a degree of self-actualisation. All of the software developers for instance saw themselves as full stack developers, and enjoyed leaning new technology and applying new knowledge and skills to their work on Harrier, despite originally

being employed to either develop software on the back-end or front-end of the stack. Furthermore, a second work identity change occurred when one developer considered themself a 'Devopeler'. This also adds support for Berg et al.'s [2010] argument that job crafting can occur beyond an individual level.

The scale of change introduced as a result of adopting DevOps may constitute a substantial challenge, and this research supports the argument from Claps et al. [2015] that organisations need to be well prepared for this. Furthermore, support is offered for Pass and Ronen's [2014] proposition that DevOps transforms the manner in which software is developed, deployed and supported through changes affecting the entire organisation. Job crafting plays a major role in these changes as they effect team roles and coordination, thus presenting a social adoption challenge [Claps et al., 2015], which must be viewed proactively by managers, rather than as an afterthought [Ghezzi, 2017].

What motivates an employee to job craft can vary considerably. While the software developers at AF expressed a desire to learn, acquire new skills and experience new technology, there was a perception that IT operations were not interested in collaborating with software developers, especially during early phases of Harrier's development. This in turn motivated software developers to further undertake perceived IT operations tasks so as to successfully automate deployment of the first Harrier builds.

For some software developers, the desire to improve performance through automation and acquire new knowledge and skills appeared to be a motivation for them job craft. As such, this finding adds support to arguments from Tims et al. [2014] that employee self-efficacy is a major motivation for job crafting, especially amongst IT workers and software developers. Furthermore, much of the job crafting appeared to be in response to the DevOps driven change at AF, therefore adding support for Mattarelli and Tagliaventi's [2012] argument that it can also be in response to change that job crafting can occur.

6.4 Conclusion and Answers to Research Questions

The disciplines of Computer Science and Software Engineering can trace their roots back to the work of 18th, 19th and 20th century pioneers including Joseph Marie Jacquard, Charles Babbage, Ada Lovelace, Alan Turing and Margaret Hamilton [Hally, 2005; Hamilton and Zeldin, 1976; Lovelace, 1843; Randell, 1994; Turing, 2009]. During the 20th century, the first methods for Software Engineering emerged in response to the increasingly unmaintainable early software, colloquially referred to as the software crisis. While the traditional approach in Software Engineering provided a structure and method for the development of software, the Agile approach was formalised at the beginning of the 21st century, with the publication of the Agile Manifesto [Beck et al., 2001]. This was in response to iterative and incremental methods to developing software, which emerged in order to tackle issues around methodological rigidity and software complexity associated with the traditional approach.

Although relatively young and perpetually changing disciplines, Computer Science and Software Engineering have contributed much. One such contribution is DevOps and this PhD research has explored what it means and how it impacts an organisation adopting it. This research has shown that to understand the adoption of DevOps, there is a need to synthesise different disciplines and in particular, apply theories and practice from Business Management, Computer Science and Software Engineering.

Defining DevOps is difficult although this research puts forward an abstract model (see figure 6.2) comprising a set of conceptual attributes (see figure 6.1) which any definition of DevOps should consider. This framework builds upon the previous work from Takimoto et al. [2016] and Smeds et al. [2015] by highlighting and including the interdisciplinary nature of DevOps. Additionally, a key component of the DevOps framework is the necessity of a continuous feedback loop. This proved critical for AF in their adoption of DevOps.

The AF case study offers a view into how software development an IT operations roles evolve in response to DevOps adoption in the organisation. While this research is contextual and therefore difficult to generalise, the results offer both a longitudinal and empirical insight, highlighting the key challenges and opportunities DevOps can bring to an organisation. Moreover, the organisational culture at AF served as an enabler to the observed job crafting. This provided a useful theoretical lens highlighting how software developers and IT operations employees reshaped their job roles, which was driven by DevOps. Two transformations of work identity was seen with software developers identifying themselves as 'full stack developers' and one calling themself a 'Devopeler' following them crafting their job further, undertaking tasks such as automating infrastructure provisioning and deployment of software, which they perceived were IT operations tasks.

Taking together the results of this research and findings from the literature, it is concluded that DevOps drives organisational change at both a socio-cultural and socio-technical level. From the case study, job crafting played a pivotal role in the changes observed with the emergence of a DevOps practice. The feedback loop depicted in figure 6.2 was critical for successful DevOps adoption in addition to being an enabler of continuous improvement and thus helping to overcome these challenges. Therefore, management must not be excluded from any view on DevOps given the intrinsic socio-cultural and socio-technical change it introduces to software development and IT operations functions. At AF, employees were not suppressed or hindered by managers when exploring and crafting their roles. Although there were organisational silos observed and some resistance to DevOps initially, the ongoing job crafting catalysed the removal of these silos. Subsequently, a collective team identity formed, with a holistic multi-functional practice and a set of shared goals.

The literature is heavily populated with the study of various tools associated with DevOps. While AF explored several tools, one in particular became a big focus. However, allowing DevOps practice to be shaped by a particular tool or platform is a critical risk, potentially coupling the process. AF discovered this following a management decision to cease using Microsoft Azure, which the DevOps practice had evolved around. It is argued that DevOps should be tool agnostic, thus it

must never be coupled to, or allowed to be determined by a specific tool.

6.4.1 Answers to Research Questions

In this section, the research questions originally posed in section 1.3, which drove this exploratory and pragmatic study on DevOps, are revisited.

1. How can DevOps be defined?
2. Why do organisations adopt DevOps?
   - **2a.** What are the perceived performance or strategic benefits?
   - **2b.** How is DevOps different to other approaches for software development?
   - **2c.** Are anticipated performance gains from its implementation realised?
3. How does DevOps adoption influence software development processes?
   - **3a.** What changes are required to the organisation and management of software development processes to enable DevOps?
4. How do software development and IT operations roles, tasks, skills, tools and work identity change as DevOps is adopted within an organisation?

6.4.1.1 How can DevOps be defined?

DevOps is difficult to universally define, although this research proposes a framework of seventeen conceptual attributes in section 6.2, which can guide in the creation of any contextual definition. These attributes are grouped into four categories: Business Driven Goals and Outcomes; Multi-Disciplinary Teams; Standardised Processes, Tools, Approaches and Principles; and Information.

However, any contextual definition must take into credence the interdisciplinary nature of DevOps, and therefore not limit it to the technical aspects of developing software and the managing business IT infrastructure.

6.4.1.2 Why do Organisations Adopt DevOps?

Both the literature and results of this research suggest that DevOps carries an economic benefit for organisations seeking to rapidly develop and deploy software of a good level of quality. In considering the overall research findings from the AF case study and the literature, it is clear that while DevOps is inherently Agile, it differs considerably from other approaches to software development. For instance, DevOps takes the Agile approach beyond the development of software by including other business functions, namely IT operations, especially with the deployment and support of software. Thus traditional IT operations tasks such as infrastructure management and systems administration become embedded into a DevOps pipeline thus, forming a constituent of the software life cycle.

There were a number of performance gains realised by AF as a result of DevOps. The rapid deployment of perceived better quality software was the most noticeable as well as empowering senior managers to address issues within critical business functions. The example of AF's invoicing department and resulting business process re-engineering highlights this impact. In this instance, DevOps enabled an approach to be taken when dealing with especially complex software requirements for a business critical function, for which the legacy system offered no solution for. The involvement of senior managers enabled issues with business process as well as legacy software to be identified and addressed as a result.

However, there are risks too, as AF discovered first hand. Allowing the formation of any dependence on particular tools must be avoided, especially where this determines process. Ultimately, the risk this presents is that DevOps becomes coupled to the use of a particular tool, meaning it can also become the single point of failure. In the case of AF, this was demonstrated with a reliance placed on the Microsoft Azure cloud platform. A management decision was taken to cease using it, leaving substantial insecurity and uncertainty for the future as a process had evolved around the use of Azure.

6.4.1.3 How does DevOps Adoption Influence Software Development Processes?

Senior managers at AF desired greater control and quality in the development of Harrier, the intended replacement for their legacy AFI software system which was developed through outsourcing. AF brought software development in-house by employing a Software Development Manager and a team of Software Developers. DevOps was emergent at AF, being driven primarily by the software developers. Senior managers allowed it to emerge which inherently provided a great deal of autonomy for the software developers.

Software development was undertaken at AF through an Agile approach, namely Scrum, with two week sprints. DevOps influenced this process by introducing the need to consider the deployment and support of software developed during each sprint and releasing it to end users. These activities involved AF's IT Operations function and two systems administrators who had always deployed any developed software and provided support to its users. As the software developers continued to explore DevOps, there was observed resistance from IT Operations to become involved.

Change is therefore intrinsic to DevOps, and impacts the wider organisation, thus necessitating a need to manage change. This sits well with the focus group and survey output where change control was agreed as being an attribute of DevOps. However, and most critically, organisations seeking to adopt DevOps must be prepared to deal with change, some of which may be seismic in scale. Bringing software development in house was a substantial change for AF. IT Operations were used to being given developed software to then deploy and support. By bringing this function together with software development, resistance and cultural issues between both teams were evident. Although kept informed, the Systems Administrators felt their input was not valued by the Software Developers, who in turn believed the former were not interested. However, both teams felt that senior management involvement in the whole process was crucial for both identifying requirements and soothing the cultural issues which emerged.

One interesting observation at AF was the reactive use of different software development methods for when requirements became especially complex. While Scrum was the primary framework utilised for developing Harrier, the invoicing department was revealed to have very complex requirements and was in a state of flux, rendering Scrum too rigid. The Software Developers desired greater flexibility to deal with this and were able to follow a Kanban framework for that department only. This fluidity in adjusting the software development framework enabled these challenges to be overcome while keeping Harrier development within a DevOps pipeline. In addition, it also led to senior management identifying significant operational issues and thus they were able to successfully re-engineer business processes for invoicing, leading to some potentially significant cost savings for the organisation.

It can be concluded that DevOps influences software development processes by facilitating adaptability in methodological approaches to developing software. Critically, DevOps can enable development activity to provide highly visible and tangible benefits to the organisation as a result. Moreover, it is argued that DevOps, while not a specific software development method in itself, is a mindset which extends beyond software developers to also include other IT functions and management by encapsulating development methods and seeking to harmonise IT, management and software development functions, with a set of shared goals. DevOps therefore requires a cultural alignment between managers, software developers and IT employees.

6.4.1.4 How do Software Development and IT Operations Roles, Tasks, Skills, Tools and Work Identity Change as DevOps is Adopted Within an Organisation?

As DevOps was adopted at AF, some changes were observed with job roles. Job crafting was prevalent with the Software Developers, with all seeing their identity at work different as a result. Initially, the Software Developers were employed to work on a particular part of the software stack, but as they continued to Job Craft in response to DevOps, they were working on both back-end and front-end

features. As time progressed, they all began to identify themselves at work as Full Stack Developers. However, a further change was also observed when one Software Developer referred to themself as a 'Devopeler', as they had also began undertaking typical IT Operations tasks, such as deployment and the provisioning of infrastructure.

One Systems Administrator was observed working more collaboratively with the Software Developers as they began to automate the provisioning of infrastructure for the Software Developers to use. While no specific work identity change was observed, the tasks being undertaken appeared synonymous with the responsibilities and tasks seen in DevOps Engineer roles (see appendix 13 on page 255).

Job crafting was also observed with a Business Analyst becoming a peacemaker between IT Operations and Software Development. Notably, this is where the biggest change in socio-culture was observed, where it went from being adversarial to collaborative. The Software Developers and Systems Administrators actively engaged and collaborated with each other following a set of shared goals. Furthermore, this extended to social activities between them outside of the workplace. One participant was also seen as a 'champion' of DevOps, as they strived to support others and carry forward a vision for what this would mean for AF.

Ultimately, these changes were a result from adopting DevOps, and by allowing both the Software Developers and Systems Administrators to engage in job crafting, senior management at AF were able to cultivate a workplace environment conducive to DevOps and had encouraged employees to explore their roles and adapt.

In answering this question, it is clear from the case study that Software Development and IT Operations job roles can change substantially due to DevOps. Yet, these changes are not necessarily negative, as shown by the benefits AF were realising. Overall, DevOps requires joined up thinking across the organisation for it to work. Therefore, this thesis concludes that DevOps is an interdisciplinary topic, from both a practical and theoretical perspective.

6.5 Theoretical Contributions

This section outlines the key contributions this PhD makes to the knowledge of DevOps. Firstly, the theoretical contributions are outlined followed by the methodological contribution this research also makes.

6.5.1 Contribution One: How to Define DevOps

Agile software development practice emerged in the late 20th century in response to the growing problems arising from traditional software development approaches. A further shift of focus came with the deployment and operation of developed software, so therefore DevOps is inherently Agile, contradicting arguments by Gupta et al. [2017] which limit Agile only to software development activity. Additionally, this research opposes the argument from Fokaefs et al. [2017], where they claim DevOps is solely focused on software development. The case study of AF presented in this thesis shows DevOps intrinsically affects the wider organisation and therefore takes the very nature of Agile beyond software development functions.

Although previous research has attempted to define DevOps [Bass et al., 2015; Cois et al., 2014; Császár et al., 2013; Dyck et al., 2015; Hosono, 2012; Mohamed, 2015; Obstfeld et al., 2014; Smeds et al., 2015; Walls, 2013], the scope and focus of these definitions differs considerably. This research explored these different definitions with practitioners and software development experts, and there was limited consensus or agreement on a single definition. However, the research did find that there are 17 core attributes (see figure 6.1), of which 6 were found to be statistically significant.

These core attributes are a key contribution of this research as they encapsulate the key aspects of DevOps, and can be used to guide organisations in developing their own context-specific definition.

A further contribution of this research is to identify that a universal definition

of DevOps should not be sought, as organisational contexts vary and therefore context-specific definitions are needed.

6.5.2 Contribution Two: Abstract Model of DevOps

The main contribution this PhD research makes builds upon the model for DevOps proposed by Takimoto et al. [2016] and was discussed in section 6.3.

Most importantly, this new model (See figure 6.2) for DevOps takes into account the core conceptual attributes agreed by practitioners including how organisational decision making, goals and strategy are both influenced by and drivers of DevOps.

This model of DevOps treats IT Operations and Software Development as specific functions which form constituents of a wider, multi-disciplinary team. As with Agile practice, DevOps is presented to be iterative but business goals ultimately drive it. Approaches, tools and processes are represented as they do act as enablers, but are by no means central to DevOps as they should be fluid and able to be adapted in response to feedback. As observed at AF, the approach for software development was able to be adjusted from Scrum to Kanban for some features at one point in response to development feedback.

Finally, the model advocates a continuous and cross-functional feedback loop, which this research argues is both central and necessary for process transparency, therefore critical to successful DevOps adoption.

6.5.3 Contribution Three: Application of Job Crafting Theory to DevOps

This research is the first to apply job crafting [Wrzesniewski and Dutton, 2001] as a theoretical lens within a software engineering environment adopting DevOps. This aided in the identification of where and why participants were actively chang-

ing the boundaries of their jobs.

The case study of AF highlights that DevOps was a driver of job crafting as it became necessary for participants to engage beyond the original remits of their roles at work. Although this research acknowledges that generalising the case study findings is difficult, the results do suggest that job crafting comes naturally to Software Developers where they are afforded the opportunity to job craft.

Additionally, job crafting offered a key theoretical insight into the observed transformation in how participants viewed their own identity and that of their colleagues at work. Within the case study, two transformations of work identity were observed, especially with software developers, where each perceived their work identity as that of a "Full Stack Developer", while one participant perceived that they had become a "Devopeler", in specific reference to how DevOps practice was leading to changes in what they do at work. It is concluded therefore that AF's adoption of DevOps enabled and encouraged the participants to job craft, in terms of their identity, relationships with their colleagues as well as their tasks and responsibilities.

6.6 Methodological Contributions

Although the theoretical contribution to the knowledge on DevOps is the most important output of this research, there are some methodological outputs too. These strengthen existing arguments and recommendations when undertaking research in a similar context or setting.

6.6.1 Contribution One: Advocation of Lethbridge et al.'s (2005) Multi-Method Recommendation

A multiple method approach to data collection is strongly argued as necessary by Lethbridge et al. [2005] when capturing information from software engineer-

ing professionals. This is to enable a researcher to capture both detailed cross sectional insights and information on process in addition to a deep longitudinal view that reveals both socio-cultural and socio-technical details.

While the longitudinal nature of this research and the open format of the diaries contributed to some participant attrition, the multi-method approach recommended by Lethbridge et al. [2005] greatly aided in mitigating this risk. Furthermore, supporting a diary study with interviews, not only produced much more data, but proved invaluable for three additional reasons.

Firstly, these interviews allowed the opportunity to probe for more details with regards to particular things arising in a participant's diary, thus resulting in a greater depth of data being collected.

Secondly, the interviews inherently acted as a means of control for the whole study, enabling participants and the researcher to discuss topics such as diary collection and how they were feeling about the research.

Finally, and linked with the second benefit, the interviews were an excellent tool in limiting any potential damage from attrition due to lack of diaries submitted.

As such, this PhD research advocates the multi-method approach recommended by Lethbridge et al. [2005], especially when undertaking any longitudinal research involving Software Developers and IT professionals.

6.6.2 Contribution Two: Utilisation of Contextual Tools for Data Collection

While the multi-method approach recommended by Lethbridge et al. [2005] was a success in its own right, this PhD research went further with specific methods of data collection.

For instance, tools such as Bitbucket, which the participants use everyday, was utilised to great effect in enabling a good continuity of data collection without

proving distracting over a 14 month period. Participants commented favourably about the manner of how diaries were collected, specifically praising how easy it was for them by using the tools they use on a daily basis.

Bitbucket enabled more frequent and detailed diary entries while aiding in the reduction of attrition, a common risk of any diary study. Additionally, the researcher flexibility was appreciated, especially for participants who did not use these tools.

The diaries were also accessible to all participants through Bitbucket and formed a method to document the process of adopting DevOps. From a research philosophical perspective, such a method of data collection sits well with the ontological view of Constructivism, where the knowledge of reality is socially constructed. While this can carry the drawback of being too open, especially where more sensitive topics would be discussed, it was mitigated by allowing participants to send such diary entries privately and directly to the researcher and away from Bitbucket.

Finally, and of benefit to AF, was the inherent knowledge management benefits this brought. As the Bitbucket repository was owned by them, it served as a very useful resource for organisational learning, enabling the reflection on change and process driven by DevOps.

In short, this PhD research advocates a Pragmatic approach to undertaking research with Software Developers and IT professionals by not just following the recommendations of Lethbridge et al. [2005], but also considering a greater openness to the manner of data collection. In particular, researchers should have some understanding of the context of their research topic and where applicable, consider utilising tools used by participants on a daily basis in order to collect data. Put simply, this means making engagement with the research process easy and flexible for participants.

6.7 Management Recommendations

As AF discovered, adopting DevOps is not straightforward. Aside from socio-cultural and socio-technical considerations localised to software development and IT functions, the impacts will be felt across the organisation. Based on the AF case study, this section outlines some practical recommendations to organisations adopting DevOps.

Firstly, and referencing Brooks's [1987] original argument, DevOps is not a silver bullet solving all software development challenges. Organisations considering DevOps should carefully appraise it at a conceptual level taking both the attributes this research proposes alongside the capabilities, cultural and technological enablers put forward by Smeds et al. [2015].

Secondly, change is intrinsic to DevOps and this research recommends management involvement to both manage the change and define business processes, especially if they change as a result. Furthermore, the AF case study provides evidence that active management involvement with DevOps is important for both being informed and driven by organisational goals as well as to inform decision making at different levels. Furthermore, the senior management involvement at AF enabled the organisation to address deficiencies in business process which the DevOps approach helped to identify.

Finally, any DevOps approach must not be determined or locked to specific tools. As AF found out through an initial reliance on Microsoft Azure, such a high degree of process and tool coupling can create a potential single point of failure.

6.8 Research Limitations

The limitations of each research method were discussed in detail within chapter 2. However, this PhD thesis acknowledges the limitations of case studies, and in particular that the findings from this research are subjective to the boundary of

the case study and as such are difficult to generalise. While offering a detailed longitudinal insight into DevOps adoption by a medium sized organisation, this case study of DevOps is ultimately unique to AF.

Furthermore, the Pragmatic and Constructivist research philosophies acknowledge the researcher has an intrinsic influence within the study. While the researcher was not embedded into the case, as would have been the case with an Ethnographic research strategy, the methodology took a multi-method approach based on recommendations from Lethbridge et al. [2005] to in a bid to capture detailed insights while mitigating risks such as retrospective bias.

6.9 Future Research

Much of the existing DevOps literature places a focus on tools. While the literature review of this thesis contained a systematic component, a meta analysis was not undertaken. Therefore, as the DevOps literature grows, further systematic reviews should be undertaken, complete with meta analyses to quantify its growth across the Business Management, Computer Science and Software Engineering disciplines and to help develop an interdisciplinary agenda to guide future research.

The research undertaken in this thesis offers a detailed and longitudinal insight of the adoption of DevOps by an organisation. While an abstract model for DevOps has been proposed, the next step would be to further test this model with small and large organisations in order to determine how generalisable it is.

The AF case study revealed DevOps to have a large business management component which would benefit from further research attention with focus on organisational behaviour and operations management. Furthermore, as a new and emerging process with the potential to produce and deploy software quickly, DevOps could be studied as a potential process innovation. A key finding from the AF case study revealed DevOps is by no means easy or straightforward for an organisation to adopt. Therefore, further work could be carried out, considering

the conceptual attributes, organisational capabilities, cultural and technical enablers [Smeds et al., 2015] to determine if an organisation is in a position where it could realistically adopt DevOps. An output of this work could potentially be a maturity model for DevOps which would link to the notions of continuous service improvement and organisational learning. Furthermore, can risk models be developed to help guide organisations through DevOps adoption?

The AF case study also revealed DevOps to inherently accelerate change in the organisation. Further research should be undertaken to understand the management implications of such rapid change driven by DevOps. While the impact of DevOps on change is a clear finding of this research, it revealed job crafting and changes to work identity occurred as well. Further study specifically on job crafting within a DevOps environment should be undertaken to consider:

- To what extent does DevOps driven change drive job crafting?
- Is job crafting an intrinsic factor in successful DevOps adoption?
- Are changes in work identity indicators of DevOps practice?

This research has provided an in-depth study of DevOps adoption by a medium sized business in the UK. However, as DevOps is a relatively new topic from both a practical and research perspective, there are many opportunities to further improve our understanding of this multi-disciplinary topic from different disciplines and organisational contexts.

References

Adams, J. (2019). Senior DevOps engineer. *Read Business Information via. LinkedIn Jobs*, available at: https://www.linkedin.com/jobs/view/1453870168/, accessed Sep 2019.

Adams, R., Jeanrenaud, S., Bessant, J., Denyer, D., and Overy, P. (2015). Sustainability-Oriented Innovation: A Systematic Review. *International Journal of Management Reviews*, pages 180–205.

Airaj, M. (2017). Enable cloud DevOps approach for industry and higher education. *Concurrency and Computation: Practice and Experience*, 29(5).

Allman, E. (2012). Managing technical debt. *Communications of the ACM*, 55(5):50–55.

Anquetil, N., de Oliveira, K. M., de Sousa, K. D., and Dias, M. G. B. (2007). Software maintenance seen as a knowledge management issue. *Information and Software Technology*, 49(5):515–529.

Barrow, P. D. M. (1999). *Investigating Stakeholder Evaluation within Rapid Application Development*. PhD thesis, School of Information Systems, University of East Anglia.

Bass, L., Jeffery, R., Wada, H., Weber, I., and Zhu, L. (2013). Eliciting operations requirements for applications. In *Proceedings of the 1st International Workshop on Release Engineering*, pages 5–8. IEEE.

Bass, L., Weber, I., and Zhu, L. (2015). *DevOps: A Software Architect's Perspective*. Addison-Wesley Professional.

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., et al. (2001). Manifesto for agile software development. available at: http://agilemanifesto.org/ accessed: Apr 2018.

Bell, T. E. and Thayer, T. A. (1976). Software Requirements: Are They Really a Problem? In *Proceedings of the 2nd international conference on Software engineering*, pages 61–68. IEEE.

Berg, J. M., Wrzesniewski, A., and Dutton, J. E. (2010). Perceiving and responding to challenges in job crafting at different ranks: When proactivity requires adaptivity. *Journal of Organizational Behavior*, 31(2-3):158–186.

Biolchini, J., Mian, P. G., Natali, A. C. C., and Travassos, G. H. (2005). Systematic Review in Software Engineering. *System Engineering and Computer Science Department COPPE/UFRJ, Technical Report ES*, 679(05).

Boehm, B. and Turner, R. (2003). *Balancing agility and discipline: A guide for the perplexed*. Addison-Wesley Professional. Indiana, USA.

Boehm, B. W. (1988). A spiral model of software development and enhancement. *Computer*, 21(5):61–72.

Bosch, J. and Bosch-Sijtsema, P. (2010). Coordination between global agile teams: From process to architecture. In Šmite, D., Moe, N. B., and Ågerfalk, P. J., editors, *Agility Across Time and Space*, pages 217–233. Springer.

Brooks, F. P. (1987). No Silver Bullet Essence and Accidents of Software Engineering. *Computer*, 20(4):10–19.

Brown, D. D. (2013). Five agile ux myths. *Journal of Usability Studies*, 8(3):55–60.

Carter, S. and Mankoff, J. (2005). When participants do the capturing: the role of media in diary studies. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 899–908. ACM.

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

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

Chilton, M. A., Hardgrave, B. C., and Armstrong, D. J. (2005). Person-job cognitive style fit for software developers: The effect on strain and performance. *Journal of Management Information Systems*, 22(2):193–226.

Claps, G. G., Svensson, R. B., and Aurum, A. (2015). On the journey to continuous deployment: Technical and social challenges along the way. *Information and Software Technology*, 57:21–31.

Clarke, J. (2011). What is a Systematic Review? *Evidence Based Nursing*, 14(3):64–64.

Cohen, J. (1968). Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological Bulletin*, 70(4):213–220.

Cois, C. A., Yankel, J., and Connell, A. (2014). Modern DevOps: Optimizing software development through effective system interactions. In *Professional Communication Conference (IPCC), 2014 IEEE International*, pages 1–7. IEEE.

Cook, N., Milojicic, D., and Talwar, V. (2012). Cloud management. *Journal of Internet Services and Applications*, 3(1):67–75.

Császár, A., John, W., Kind, M., Meirosu, C., Pongrácz, G., Staessens, D., Takacs, A., and Westphal, F.-J. (2013). Unifying cloud and carrier network: Eu fp7 project unify. In *Utility and Cloud Computing (UCC), 2013 IEEE/ACM 6th International Conference on*, pages 452–457. IEEE.

de Vasconcelos, J. B., Kimble, C., Carreteiro, P., and Rocha, Á. (2017). The application of knowledge management to software evolution. *International Journal of Information Management*, 37(1):1499–1506.

De Vaus, D. (2013). *Surveys in social research*. Routledge, 6th edition. Abingdon.

Dempster, M. (2011). *A Research Guide for Health and Clinical Psychology*. Palgrave Macmillan.

Dennehy, D. and 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.

DeVellis, R. F. (2016). *Scale development: Theory and applications*, volume 26. Sage, 4th edition.

Diamond, M., Allcorn, S., and Stein, H. (2004). The surface of organizational boundaries: A view from psychoanalytic object relations theory. *Human Relations*, 57(1):31–53.

Dijkstra, E. W. (1972). The Humble Programmer. *Communications of the ACM*, 15(10):859–866.

Dyck, A., Penners, R., and Lichter, H. (2015). Towards definitions for release engineering and DevOps. In *Release Engineering (RELENG), 2015 IEEE/ACM 3rd International Workshop on*, pages 3–3. IEEE.

Eisenhardt, K. M. (1989). Building theories from case study research. *Academy of management review*, 14(4):532–550.

Eisenhardt, K. M. and Graebner, M. E. (2007). Theory building from cases: Opportunities and challenges. *Academy of management journal*, 50(1):25–32.

Emerson, R. M., Fretz, R. I., and Shaw, L. L. (2011). *Writing ethnographic fieldnotes*. University of Chicago Press. Chicago: IL.

Erich, F., Amrit, C., and Daneva, M. (2014). A mapping study on cooperation between information system development and operations. In *15th International Conference on Product-Focused Software Process Improvement (PROFES) 2014, Helsinki, Finland*, pages 277–280. Springer.

Eveleens, J. L. and Verhoef, C. (2010). The rise and fall of the chaos report figures. *IEEE software*, 27(1):30–36.

Feitelson, D. G., Frachtenberg, E., and Beck, K. L. (2013). Development and deployment at Facebook. *IEEE Internet Computing*, 17(4):8–17.

Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures*. PhD thesis, University of California, Irvine Doctoral dissertation.

Fitzgerald, B. and Stol, K.-J. (2017). Continuous software engineering: A roadmap and agenda. *Journal of Systems and Software*, 123:176–189.

Fokaefs, M., Barna, C., and Litoiu, M. (2017). From DevOps to BizOps: Economic sustainability for scalable cloud applications. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 12(4):25.

Friedman, A. L. and Cornford, D. S. (1989). *Computer Systems Development: History Organization and Implementation*. John Wiley & Sons, Inc.

Fuegi, J. and Francis, J. (2003). Lovelace & babbage and the creation of the 1843'notes'. *IEEE Annals of the History of Computing*, 25(4):16–26.

Geertz, C. (1973). *The interpretation of cultures*. Basic Books. New York.

Ghezzi, C. (2017). Of software and change. *Journal of Software: Evolution and Process*, 29(9).

Gill, J. and Johnson, P. (2010). *Research methods for managers*. Sage, 4th edition. London.

Glaser, B. and Strauss, A. L. (2017). *Discovery of grounded theory: Strategies for qualitative research*. Routledge. Oxon.

Glass, R. L. (2006). The standish report: does it really describe a software crisis? *Communications of the ACM*, 49(8):15–16.

Gohil, K., Alapati, N., and Joglekar, S. (2011). Towards behavior driven operations (bdops). In *Advances in Recent Technologies in Communication and Computing (ARTCom 2011), 3rd International Conference on*, pages 262–264. IEEE.

Greenhalgh, T., Peacock, R., et al. (2005). Effectiveness and Efficiency of Search Methods in Systematic Reviews of Complex Evidence: Audit of Primary Sources. *BMJ*, 331(7524):1064–1065.

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

Gupta, V., Kapur, P., and Kumar, D. (2017). Modeling and measuring attributes influencing DevOps implementation in an enterprise using structural equation modeling. *Information and Software Technology*, 92:75–91.

Hally, M. (2005). *Electronic brains: stories from the dawn of the computer age*. Joseph Henry Press. Washington DC.

Hamilton, M. (2018). The Language as a Software Engineer. available at: https://www.youtube.com/watch?v=ZbVOF0Uk5lU accessed: May 2020.

Hamilton, M. and Zeldin, S. (1976). Higher order software—a methodology for defining software. *IEEE Transactions on Software Engineering*, SE-2(1):9–32.

Hanley, T. and Cutts, L. (2013). What is a Systematic Review? *Counselling Psychology Review*, 28(4):3–6.

Hollings, C., Martin, U., and Rice, A. (2018). *Ada Lovelace: The Making of a Computer Scientist*. Bodleian Library, University of Oxford. Oxford, UK.

Hosono, S. (2012). A DevOps framework to shorten delivery time for cloud applications. *International Journal of Computational Science and Engineering*, 7(4):329–344.

Hussaini, S. W. (2014). Strengthening harmonization of development (dev) and operations (ops) silos in it environment through systems approach. In *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*, pages 178–183. IEEE.

Ilgen, D. R. and Hollenbeck, J. (1991). Job design and roles. In Dunnette, M. and Hough, L., editors, *Handbook of Industrial and Organizational Psychology*, pages 165–207. Consulting Pyschologists Press.

Ince, D. (1988). *Software development: Fashioning the baroque*. Oxford University Press, Inc.

Jesson, J., Matheson, L., and Lacey, F. M. (2011). *Doing your Literature Review: Traditional and Systematic Techniques*. Sage. London.

Jones, S., Noppen, J., and Lettice, F. (2016). Management Challenges for DevOps Adoption within UK SMEs. In *Proceedings of the 2nd International Workshop on Quality-Aware DevOps, July 21 2016, Saarbrücken, Germany*, pages 7–11. ACM.

Jørgensen, M. and Moløkken-Østvold, K. (2006). How large are software cost overruns? a review of the 1994 chaos report. *Information and Software Technology*, 48(4):297–301.

Kahn, R. L. and Cannell, C. F. (1957). *The dynamics of interviewing; theory, technique, and cases.* Wiley. Oxford, UK.

Kantsev, V. (2017). *Implementing DevOps on AWS*. Packt Publishing Ltd.

Karl, H., Dräxler, S., Peuster, M., Galis, A., Bredel, M., Ramos, A., Martrat, J., Siddiqui, M. S., Van Rossem, S., Tavernier, W., et al. (2016). DevOps for network function virtualisation: an architectural approach. *Transactions on Emerging Telecommunications Technologies*, 27(9):1206–1215.

Kim, M., Mohindra, A., Muthusamy, V., Ranchal, R., Salapura, V., Slominski, A., and Khalaf, R. (2016). Building scalable, secure, multi-tenant cloud services on ibm bluemix. *IBM Journal of Research and Development*, 60(2-3):8:1–8:12.

Kitchenham, B. (2004). Procedures for Performing Systematic Reviews. *Keele, UK, Keele University*, 33(2004):1–26.

Kitchenham, B., Brereton, O. P., Budgen, D., Turner, M., Bailey, J., and Linkman, S. (2009). Systematic literature reviews in software engineering - A systematic literature review. *Information and software technology*, 51(1):7–15.

Kneuper, R. (2017). Sixty years of software development life cycle models. *IEEE Annals of the History of Computing*, 39(3):41–54.

Knox, K. (2004). A researcher's dilemma-philosphical and methodological pluralism. *The Electronic Journal of Business Research Methods*, 2(2):119–128.

Koilada, D. K. (2019). Business model innovation using modern devops. In *2019 IEEE Technology Engineering Management Conference (TEMSCON)*, pages 1–6.

Landis, J. R. and Koch, G. G. (1977). The Measurement of Observer Agreement for Categorical Data. *Biometrics*, 33(1):159–174.

Lapham, M. A. (2014). Software sustainment–now and future. *CrossTalk*, 27(1):33–36.

Lawley, D. N. and Maxwell, A. E. (1962). Factor analysis as a statistical method. *Journal of the Royal Statistical Society. Series D (The Statistician)*, 12(3):209–229.

Lethbridge, T. C., Sim, S. E., and Singer, J. (2005). Studying software engineers: Data collection techniques for software field studies. *Empirical software engineering*, 10(3):311–341.

Liu, Y., Li, C., and Liu, W. (2014). Integrated solution for timely delivery of customer change requests: A case study of using DevOps approach. *International Journal of U-& E-Service, Science & Technology*, 7(2):41–50.

Loukides, M. (2012). *What is DevOps?* O'Reilly. Sebastopol, USA.

Lovelace, A. (1843). Notes by AAL (augusta ada lovelace). *Taylor's Scientific Memoirs*, pages 666–731. *Lovelace's translation of Menabrea together with her "Notes" are also available online at:* http://www.fourmilab.ch/babbage/sketch.html .

Lwakatare, L. E., Kilamo, T., Karvonen, T., Sauvola, T., Heikkilä, V., Itkonen, J., Kuvaja, P., Mikkonen, T., Oivo, M., and Lassenius, C. (2019). Devops in practice: A multiple case study of five companies. *Information and Software Technology*, 114:217–230.

Mäkikangas, A., Bakker, A. B., and Schaufeli, W. B. (2017). Antecedents of daily team job crafting. *European Journal of Work and Organizational Psychology*, 26(3):421–433.

Martini, A., Pareto, L., and Bosch, J. (2013). Improving businesses success by managing interactions among agile teams in large organizations. In *International Conference of Software Business*, pages 60–72. Springer.

Mattarelli, E. and Tagliaventi, M. R. (2012). How offshore professionals' job dissatisfaction can promote further offshoring: Organizational outcomes of job crafting. *Journal of Management Studies*, 52(5):585–620.

McLarnon, B., Robinson, P., Milligan, P., and Sage, P. (2014). An iterative approach to trustable systems management automation and fault handling. *Journal of Network and Systems Management*, 22(3):366–395.

Mohamed, S. I. (2015). DevOps shifting software engineering strategy Value based perspective. *ISOR Journal of Computing Engineering (ISOR-JCE)*, 17(2):51–57.

Moore, G. E. (1965). Cramming more components onto integrated circuits. *Electronics*, pages 114–117.

Obstfeld, J., Knight, S., Kern, E., Wang, Q. S., Bryan, T., and Bourque, D. (2014). Virl: the virtual internet routing lab. In *Proceedings of the 2014 ACM conference on SIGCOMM*, pages 577–578. ACM.

Ohly, S., Sonnentag, S., Niessen, C., and Zapf, D. (2010). Diary studies in organizational research. *Journal of Personnel Psychology*, 9(2):79–93.

Oliveira, F., Eilam, T., Nagpurkar, P., Isci, C., Kalantar, M., Segmuller, W., and Snible, E. (2016). Delivering software with agility and quality in a cloud environment. *IBM Journal of Research and Development*, 60(2-3):10:1–10:11.

Pass, S. and Ronen, B. (2014). Reducing the software value gap. *Communications of the ACM*, 57(5):80–87.

Piekkari, R., Welch, C., and Paavilainen, E. (2009). The case study as disciplinary convention: Evidence from international business journals. *Organizational research methods*, 12(3):567–589.

Platt, J. (1992). "case study" in american methodological thought. *Current Sociology*, 40(1):17–48.

Poppleton, S., Briner, R. B., and Kiefer, T. (2008). The roles of context and everyday experience in understanding work-non-work relationships: A qualitative diary study of white-and blue-collar workers. *Journal of Occupational and Organizational Psychology*, 81(3):481–502.

Qumer, A. and Henderson-Sellers, B. (2008). A framework to support the evaluation, adoption and improvement of agile methods in practice. *Journal of Systems and Software*, 81(11):1899–1919.

Ragin, C. C. (1992). Introduction: cases of" what is a case?"(pp. 1-17). In Ragin, C. C. and Becker, H. S., editors, *What is a case?: Exploring the foundations of social inquiry*, pages 1–17. Cambridge University Press, Cambridge.

Ragin, C. C. (1997). Turning the tables: How case-oriented research challenges variable-oriented research. *Comparative social research*, 16:27–42.

Ranchal, R., Mohindra, A., Manweiler, J. G., and Bhargava, B. (2015). RADical strategies for engineering web-scale cloud solutions. *IEEE Cloud Computing*, 2(5):20–29.

Randell, B. (1994). The origins of computer programming. *IEEE Annals of the History of Computing*, 16(4):6–14.

Randell, B. (1996). The 1968/69 NATO Software Engineering Reports. *History of Software Engineering*, page 37.

Rayl, A. J. S. (2008). NASA Engineers and Scientists-Transforming Dreams Into Reality. available at: https://www.nasa.gov/50th/50th_magazine/scientists.html accessed: May 2020.

Reis, H. T. and Gable, S. L. (2000). Event-sampling and other methods for studying everyday experience. In Reis, H. T. and Judd, C. M., editors, *Handbook of research methods in social and personality psychology*, pages 190–222. Cambridge University Press, Cambridge.

Rhodes, C. (2016). Business Statistics. *House of Commons Library*, available at: http://researchbriefings.files.parliament.uk/documents/SN06152/SN06152.pdf, accessed: Mar 2017.

Robson, C. and McCartan, K. (2016). *Real World Research*. Wiley, 4th edition. Chichester.

Roche, J. (2013). Adopting DevOps practices in quality assurance. *Communications of the ACM*, 56(11):38–43.

Royce, W. W. (1970). Managing the development of large systems: Concepts and techniques. In *9th International Conference on Software Engineering*, pages 328–38. ACM.

Rubin, K. S. (2012). *Essential Scrum: A practical guide to the most popular Agile process*. Addison-Wesley. Michigan, USA.

Saunders, M. N., Saunders, M., Lewis, P., and Thornhill, A. (2011). *Research Methods for Business Students*. Prentice Hall, 5th edition. Harlow.

Sebastian, I. M., Ross, J. W., Beath, C., Mocker, M., Moloney, K. G., and Fonstad, N. O. (2017). How big old companies navigate digital transformation. *MIS Quarterly Executive*, 16(3):197–213.

Sill, A. (2015). Emerging standards and organizational patterns in cloud computing. *IEEE Cloud Computing*, 2(4):72–76.

Simons, H. (2009). *Case study research in practice*. SAGE publications.

Smeds, J., Nybom, K., and Porres, I. (2015). DevOps: a definition and perceived adoption impediments. In *International Conference on Agile Software Development*, pages 166–177. Springer.

Smidts, C., Stutzke, M., and Stoddard, R. W. (1998). Software reliability modeling: an approach to early reliability prediction. *IEEE Transactions on Reliability*, 47(3):268–278.

Sommerville, I. (1992). *Software Engineering*. Addison-Wesley, 4th edition. New York.

Stake, R. E. (1994). Case studies. In Denzin, N. K. and Lincoln, Y. S., editors, *Handbook of qualitative research*, pages 236–247. Sage, Thousand Oaks, CA.

Stake, R. E. (1995). *The art of case study research*. Sage, Thousand Oaks, CA.

Stake, R. E. (2000). Case studies. In Denzin, N. K. and Lincoln, Y. S., editors, *Handbook of qualitative research*, pages 435–454. Sage, Thousand Oaks, CA, 2nd edition.

Suddaby, R. (2006). From the editors: What grounded theory is not. *Academy of Management Journal*, 49(4):633–642.

Sun, D., Fu, M., Zhu, L., Li, G., and Lu, Q. (2016). Non-intrusive anomaly detection with streaming performance metrics and logs for DevOps in public clouds: a case study in aws. *IEEE Transactions on Emerging Topics in Computing*, 4(2):278–289.

Takimoto, M., Komine, H., and Tamura, K. (2016). Network DevOps solution for creating new network services. *FUJITSU Sci. Tech. J*, 52(2):8–12.

Tamburri, D. A., Kruchten, P., Lago, P., and van Vliet, H. (2015). Social debt in software engineering: insights from industry. *Journal of Internet Services and Applications*, 6(1):1–17.

Teddlie, C. (2005). Methodological issues related to causal studies of leadership: A mixed methods perspective from the usa. *Educational Management Administration & Leadership*, 33(2):211–227.

Thomas, G. (2011). A typology for the case study in social science following a review of definition, discourse, and structure. *Qualitative inquiry*, 17(6):511–521.

Tims, M., Bakker, A. B., and Derks, D. (2014). Daily job crafting and the self-efficacy - performance relationship. *Journal of Managerial Psychology*, pages 490–507.

Tranfield, D. R., Denyer, D., and Smart, P. (2003). Towards a methodology for developing evidence-informed management knowledge by means of systematic review. *British Journal of Management*, 14:207–222.

Tseitlin, A. (2013). The antifragile organization. *Communications of the ACM*, 56(8):40–44.

Turing, A. M. (1937). On Computable Numbers, with an Application to the Entscheidungsproblem. *J. of Math*, 58:345–363.

Turing, A. M. (2009). Computing Machinery and Intelligence. In *Parsing the Turing Test*, pages 23–65. Springer. *Originally published by Turing in 1950.*

Veres, O., Kunanets, N., Pasichnyk, V., Veretennikova, N., Korz, R., and Leheza, A. (2019). Development and operations - the modern paradigm of the work of it project teams. In *2019 IEEE 14th International Conference on Computer Sciences and Information Technologies (CSIT)*, volume 3, pages 103–106.

Walls, M. (2013). *Building a DevOps Culture*. O'Reilly. Sebastopol, USA.

Welch, C., Piekkari, R., Plakoyiannaki, E., and Paavilainen-Mäntymäki, E. (2011). Theorising from case studies: Towards a pluralist future for international business research. *Journal of International Business Studies*, 42(5):740–762.

Wettinger, J., Breitenbücher, U., Falkenthal, M., and Leymann, F. (2017). Collaborative gathering and continuous delivery of DevOps solutions through repositories. *Computer Science-Research and Development*, 32(3-4):281–290.

Wettinger, J., Breitenbücher, U., Kopp, O., and Leymann, F. (2016). Streamlining DevOps automation for cloud applications using TOSCA as standardized metamodel. *Future Generation Computer Systems*, 56:317–332.

Wieviorka, M. (1992). Case studies: History or sociology. In Ragin, C. and S, B. H., editors, *What is a case? Exploring the foundations of social inquiry*, pages 159–172. Cambridge University Press, Cambridge.

Wrzesniewski, A. and Dutton, J. E. (2001). Crafting a job: Revisioning employees as active crafters of their work. *Academy of Management Review*, 26(2):179–201.

Yin, R. K. (2013). *Case study research: Design and methods*. Sage, 5th edition.

Young, R. A. and Collin, A. (2004). Introduction: Constructivism and social constructionism in the career field. *Journal of vocational behavior*, 64(3):373–388.

Zikmund, W. G., Babin, B. J., Carr, J. C., and Griffin, M. (2013). *Business Research Methods*. Cengage Learning, 9th edition. Mason, OH.

Šmite, D., Moe, N. B., and Ågerfalk, P. J. (2010). Coordination between global agile teams: From process to architecture. In *Agility Across Time and Space*. Springer.

Appendices

Appendix 1: Focus Group Itinerary

09:00 Registration, Tea/Coffee and Networking.

09:15 Welcome and Overview (S Jones / F Lettice / J Noppen).

09:20 Exercise 1 – Discuss and agree on conceptual attributes of DevOps in sub-groups.

10:00 Feedback and discussion with whole group.

10:20 Tea/Coffee and Networking.

10:40 Exercise 2 – Evaluate/Produce definitions based on agreed conceptual in sub-groups.

11:30 Feedback and discussion with whole group – agree on final definition.

12:00 Endnote: "Hi, I'm a Devopeler!" (Dom Davis).

12:15 Q&A, Closing Remarks and Finish (S Jones / F Lettice / J Noppen).

12:25 Lunch and Networking.

Appendix 2: Focus Group Photos

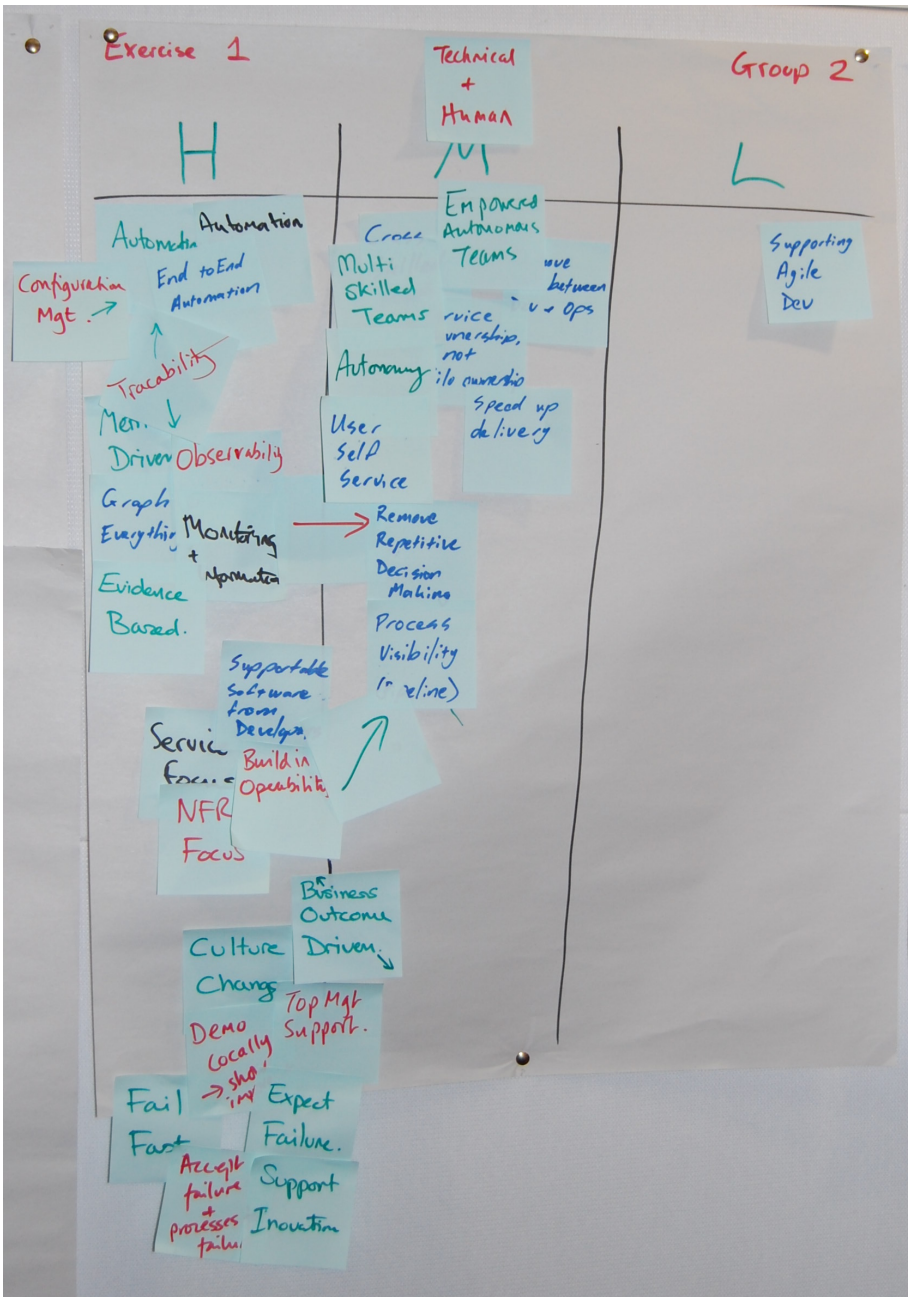


Figure 6.5: Example of a set of prioritised attributes of DevOps as undertaken by participants within group 2.

Figure 6.6: Researcher placing DevOps definitions on the wall.

Appendix 3: Specimen Questionnaire

DevOps: Towards an industry grounded definition.

This work seeks to examine the ongoing problem of defining DevOps, and forms a major constituent of a doctoral research project at the University of East Anglia. This very short survey seeks to capture the opinion on two definitions produced in conjunction with industrial partners actively practising DevOps.

The survey is structured into four main sections and should take no more than 10 minutes to complete.

If you have any questions regarding this survey, the nature of the research being conducted, would like to know more or are potentially interested in participating in further DevOps research, please contact:

- Steve Jones (primary researcher): stephen.j.jones(at)uea.ac.uk
- Fiona Lettice (supervisor): f.lettice(at)uea.ac.uk
- Joost Noppen (supervisor): j.noppen(at)uea.ac.uk.

Many thanks for your time and consideration in completing this short survey.

*Required

1. You and your views on DevOps

This section seeks to know more about the organisation you work for, and your agreement on concepts potentially related to DevOps.

1. **1.1 How old is the organisation/business your work for?** *
   Mark only one oval.
   - ◯ 0 - 3 years
   - ◯ 4 - 9 years
   - ◯ 10 - 20 years
   - ◯ 21 - 50 years
   - ◯ 50 or more years
   - ◯ Don't Know

2. **1.2 How many employees does the organisation/business you work for have?** *
   Mark only one oval.
   - ◯ 0 - 10
   - ◯ 11 - 20
   - ◯ 21 - 50
   - ◯ 51 - 100
   - ◯ 101 - 249
   - ◯ 250 or more
   - ◯ Don't Know

3. **1.3 What sector is the organisation/business within?** *
Mark only one oval.

- ◯ Private
- ◯ Public
- ◯ Non-Profit/Charity
- ◯ Don't Know

4. **1.4 Why does the organisation/business develop software?** *
Mark only one oval.

- ◯ For profit
- ◯ For internal use
- ◯ Both of the above

5. **1.5 In your opinion, is DevOps a methodology? role? both? or something else?** *
Mark only one oval.

- ◯ Methodology
- ◯ Role
- ◯ Both
- ◯ Don't Know
- ◯ Other: _____

6. **1.6 Do you agree or disagree that the following approaches/concepts are important in DevOps?** *
Mark only one oval per row.

	Strongly Agree	Agree	Neither	Disagree	Strongly Disagree
Automation	◯	◯	◯	◯	◯
Change Control	◯	◯	◯	◯	◯
Configuration Management	◯	◯	◯	◯	◯
Data Analytics	◯	◯	◯	◯	◯
Service Management	◯	◯	◯	◯	◯

7. **1.7 Do you agree or disagree that the following team aspects are important in DevOps?** *
Mark only one oval per row.

	Strongly Agree	Agree	Neither	Disagree	Strongly Disagree
Accountability	◯	◯	◯	◯	◯
Decision Making	◯	◯	◯	◯	◯
Ownership	◯	◯	◯	◯	◯
Responsibility	◯	◯	◯	◯	◯
Skills	◯	◯	◯	◯	◯

8. **1.8 Do you agree or disagree that the following business outcomes are important in DevOps?** *
   Mark only one oval per row.

	Strongly Agree	Agree	Neither	Disagree	Strongly Disagree
Reduced cost					
Informed Decision Making					
Quality					
Simplicity/Granularity					
Time					

9. **1.9 Do you agree or disagree that the following information aspects are important in DevOps?** *
   Mark only one oval per row.

	Strongly Agree	Agree	Neither	Disagree	Strongly Disagree
Measurability / Metrics					
Observability					

2. DevOps - Definition Evaluation

Please examine the following two definitions and proceed to answer the question below indicating your preference.

Definition 1

"DevOps is a continuous improvement methodology that uses a set of tools, streamlined and automated processes, and empowered, multi-disciplinary teams to deliver, operate and inform business outcomes."

Definition 2

"DevOps is an evolution in how IT services are delivered and supported. It stresses cross functional collaboration to bridge the gradational organisational process divide between development and operational teams. It aims to reduce the time between committing a change to a system and the change being placed into production."

10. **2.1 Which definition do you prefer?** *
    Mark only one oval.

    - Definition 1
    - Definition 2

3. Your Comments on the Definitions

In the next two questions, please provide up to three positive and three negative comments for each definition. This can be as simple as list of words, or paragraphs if you prefer. The definitions are repeated below for your convenience:

Definition 1

"DevOps is a continuous improvement methodology that uses a set of tools, streamlined and automated processes, and empowered, multi-disciplinary teams to deliver, operate and inform business outcomes."

11. **3.1 - Definition 1 - Positives** *

12. **3.2 - Definition 1 - Negatives** *

Definition 2

"DevOps is an evolution in how IT services are delivered and supported. It stresses cross functional collaboration to bridge the gradational organisational process divide between development and operational teams. It aims to reduce the time between committing a change to a system and the change being placed into production."

13. **3.3 - Definition 2 - Positives** *

14. **3.4 - Definition 2 - Negatives** *

4. Nearly there...

Just a few more questions, mainly focusing on your role, and how it fits into DevOps.

15. **4.1 What is your current job title?** *

Appendix 4: Diary Template and Guide Questions

Markdown Diary Template

```
# DevOps Study - AF and UEA
_Diary Kept by: <your name>_
***

## Entry Title
Date: _[Date of Diary Entry]_
<content here>
***

## Entry Title
Date: _[Date of Diary Entry]_
<content here>
***

## Entry Title
Date: _[Date of Diary Entry]_
<content here>
***
```

Plain Text Diary Template

```
DevOps Study - AF and UEA
Diary Kept by: <your name>

-----------------------------------------------

<Entry Title>
Date: <date>
<content here>

-----------------------------------------------

<Entry Title>
Date: <date>
<content here>

-----------------------------------------------

<Entry Title>
Date: <date>
<content here>
```

Guide Questions for Diary Study

Participants are encouraged to consider these questions in their reflective diaries, but not to necessarily answer each question in every entry.

- What are you doing differently because of the new DevOps approach being taken?
  - Have you changed the way in which you do your tasks?
  - Have you taken on any new tasks or projects?
  - Have you changed the way in which you see your role in the organisation?
  - Have you taken the initiative to work with different people in the organisation?
- What has been happening at work today/this week?
  - What went well at work today/this week and why?
  - What have been the key challenges at work today/this week and why?
  - Any breakthroughs, personal and/or team achievements?
- How is the DevOps approach smoother/better, bumpier/worse or no different than previous approaches to software development you've used?
  - How does it compare to agile processes (e.g. Scrum)?
  - How does it compare to traditional non-agile processes (e.g. Waterfall)?
- What do you think you or your team, manager(s) and or organisation as a whole need to do or learn to make the DevOps approach work better?

Appendix 5: Protocol for Entrance Interviews

1. **(Open)** You've given your job title as <job title>, can you tell me more about what you do in this role at <organisation>?
    - **a.** ***(Probe)*** What is the most enjoyable thing about working for <organisation>?
    - **b.** ***(Probe)*** What is the most challenging part of your role here?
    - **c.** ***(Closed) (Managers)*** is this your first management role?
        - i. ***(If no)*** When did you first manage other people?

Notes for Question 1:
Consider paraphrasing a listed output and be sure to check the individuals role.

2. **(Open)** What did you do prior to joining <organisation>?
    - **a.** ***(Probe)*** How different was this role to the one you now do here?

Notes for Question 2:
Consider splitting the sub question if necessary to consider roles in different firms.

3. **(Closed)** Going back to your job role at <organisation>, are you aware that <organisation> is adopting "DevOps"?
    - ***a.*** ***Probe:*** What does "DevOps" mean to you?

Notes for Question 3a:
The aim is to probe what knowledge they possess about DevOps, this may need additional thinking on the spot with sharp follow-up questions to answers.

- ***b.*** ***(Closed)*** Have you worked in a DevOps environment previously?
    - i. ***(Probe) (If Yes)*** How experienced do you feel with regards to DevOps?
    - ii. ***(Probe) (If Yes)*** How does it compare to the DevOps approach being adopted by <organisation>?
    - iii. ***(Probe) (Both)*** *How do you feel about <organisation>'s adoption of DevOps?*
    - iv. ***(Probe) (Managers)*** How does management differ between DevOps and non-DevOps approaches?

- ***c.*** ***(Closed)*** **(Managers)** Do you feel your developers and operations staff require additional skills for successfully working in a DevOps environment?
    - i. ***(Probe)*** What skills in particular?
    - ii. ***(Probe)*** What skills do they already possess?
    - iii. ***(Probe)*** When recruiting Developers/IT Operations, what skills and experience do you look for?

Notes for Questions 3b and 3c:
Interesting to glean if the manager feels DevOps is a ***role*** *or* ***approach****. Only ask iii if they are actively involved in recruitment, and also be careful not to "lead" with this question though.*

- ***d.*** ***(Open)*** What tools would you associate with DevOps?
    - i. ***(Probe)*** What does <tool> do?
    - ii. ***(Closed) (Managers)*** Does <organisation> have any preference to open source or propriety tools?

Notes for Question 3d:
Consider paraphrasing a listed output. Ask them to give an overview of specific tools as a means to seeing what they actually know.

e. ***(Open)*** What do you believe Development Staff/IT Operations (ask as appropriate) staff to do in their day to day job roles in a DevOps environment?
   i. ***(Probe)*** How does this differ from a non-DevOps environment?
   ii. ***(Probe)*** What (if any) cultural barriers exist by your perception?
   iii. ***(Probe) (Managers)*** What management challenges does this raise?

Notes for Question 3e:
This question needs to be targeted at opposites… i.e. Devs need to be asked about Ops and Vice Versa. Managers can be asked about both, especially if they are not specific to say Dev or Ops themselves.

4. ***(Open)*** What does the notion of "Infrastructure as Code" mean to you?
   a. ***(Probe)*** Who is it relevant too? IT Operations? Developers? Management?
   b. ***(Probe) (If unknown / don't know)*** *What do you perceive it to be / mean?*

Notes for Question 4:
Do NOT explain what is meant by "infrastructure as code" if the participant doesn't know. Instead, ask them to provide their perception or what they believe it could mean on face value. If they don't know, do not spend long on this question.

5. ***(Closed)*** Do you see your role changing as <organisation> further embraces DevOps?
   a. ***(Probe)(if yes)*** In what ways do you perceive it changing?
   b. ***(Probe)*** Do you see the culture at <organisation> changing too, if so, in what ways?
   c. ***(Probe)*** Who is responsible for software deployment and maintenance at <organisation>?

6. ***(Open)*** In what ways do you perceive DevOps is bringing or will bring change to <organisation>?
   a. ***(Probe)*** How do you feel such change would affect you?
   b. ***(Probe) (Managers)*** Who would be affected by any change?
   c. ***(Probe) (Managers)*** How well do you feel <organisation> copes with change?
   d. ***(Probe) (Managers)*** What (if any) strategies do you have with regards to handing DevOps related change?
   e. ***(Probe)*** *Where do you perceive the impact of DevOps associated change will occur within <organisation>?*
   f. ***(Probe) (Developers)*** What methods did <organisation> use previously for Software Development?
   g. ***(Probe)(IT Operations)*** How was infrastructure managed at <organisation> prior to DevOps adoption?

Notes for Questions 5 and 6:
Questions 5 and 6 may well be answered in tandem, so be aware. It would also be good to gauge perceptions too.

7. ***(Open)*** What is the perceived benefit(s) that DevOps will bring to <organisation>?

Notes for Question 7:
This is a critical question – be sure to at least ask this before finishing.

The final three questions are relaxed, and are designed to allow the participant an opportunity to directly raise any questions or concerns with the research in the privacy of the interview.

8. How do you feel with regards to this research?

9. Do you have any questions or concerns regarding the research project?

10. *Do you have any other questions or things you wish to discuss?*

Notes for Question:
Questions 8 and 9 and 10 are a chance for the participant to ask any question they may have and to find out further information.

End of Interview checklist:

- Thank the participant for their time and insights.
- Reiterate the options on the consent form – in particular, double check if they wish to receive a transcript or the recording of the interview.
- Outline the next steps…
  - Diary Study
  - Transcriptions
  - Mid-Term Interviews
  - Exit Interview
- Ensure participant has contact details (provide a business card if possible).

Appendix 6: Protocol for Mid-Study Interviews

1. What does DevOps mean to you?
    a. ***(Probe)*** How has <organisation> has taken a DevOps approach?
        i. Have you noticed any improvements? If so, what?
    b. ***(Probe)*** In your opinion, describe the role that senior management play in DevOps?

Notes for Question 1:

Capture views on DevOps and its adoption. Probe for discussion.

2. **(Open)** In your opinion, how has development progressed since the last interview?
    a. ***(Probe)*** What has been the most challenging aspects?
    b. ***(Probe)*** What has gone well?
    c. ***(Probe)*** What, if anything, has gone wrong?
    d. ***(Closed) (Managers)*** What, if any, management challenges have arisen?
        i. How have you mitigated these challenges?

Notes for Question 2:

Capture individuals perceptions on the progress. Probe for discussion.

3. **(Closed)** Are you still having to maintain <legacy system>?
    a. ***(Probe)*** What challenges does this introduce, if any?
    b. ***(Probe)*** How would things be different if you did not have to deal with <legacy system>?
    c. ***(Probe)(Manager)*** How many developer hours, on average, are being taken up with work on <legacy system>?
        i. How does this impact on Harrier development?

Notes for Question 3:

Probing the the impact of legacy system maintenance on development activities where applicable. This is especially relevant for Anglia Farmers, given the AFI system and the potential impact to Harrier development.

Probing of participant's diary entries.

Discuss themes and topics of interest arising from the participants submitted diaries

---

The final three questions are relaxed, and are designed to allow the participant an opportunity to directly raise any questions or concerns with the research in the privacy of the interview.

4. How do you feel with regards to this research?

5. Do you have any questions or concerns regarding the research project?

6. *Do you have any other questions or things you wish to discuss?*

Notes for Questions 4, 5 and 6:
Questions 5 and 6 and 7 are a chance for the participant to ask any question they may have and to find out further information.

End of Interview checklist:

- Thank the participant for their time and insights.
- Reiterate the options on the consent form – in particular, double check if they wish to receive a transcript or the recording of the interview.
- Outline the next steps…
  - Diary Study
  - Transcriptions
  - Mid-Term Interview 2 (for Anglia Farmers)
  - Exit Interview
- Ensure participant has contact details (provide a business card if possible).

Appendix 7: Protocol for Exit Interviews

1. What does DevOps mean to you?
    a. ***(Probe)*** How has <organisation> has achieved DevOps?
    b. **(Probe)** Where did you fit in to the DevOps approach?
    c. **(Probe)** What has this meant for <new system development> and <legacy system maintenance> (if applicable)?
    d. **(Probe)** Has anything gone wrong, if so, what and why?
    e. **(Probe)** How often is <organisation> releasing new features into production?
        i. **(Probe)** How has this changed since <start of study period>?
    f. **(Probe)** How does management feature in <organisation's> DevOps approach?

Notes for Question 1:
Capture views on DevOps and the adoption of it at the organisation. This needs to ask how it has achieved this. In particular, probe legacy system maintenance with AF participants.

2. **(Closed)** Has your role changed since <start of study period>?
    a. ***(Probe)*** **(If yes)** How and in what ways?
    b. **(Probe)** How has <Software Development> / <IT operations> function/team changed since <start of study period>?
    c. **(Probe)** Has their been any integration with <other function> in terms of DevOps and if so, in what ways?

Notes for Question 2:
About probing the work identity – how has this changed? Also look into their perceptions of the team – has it changed? How?. Probe for function/team level changes – especially transformative.

Probing of participant's diary entries.

Discuss themes and topics of interest arising from the participants submitted diaries

---

3. **(Open) How has Working for <Organisation> compared to working in your previous role(s)?**

4. **(Close) Do you feel your time at <Organisation> has helped if you go into a DevOps environment elsewhere?**
    a. ***(Probe)*** In what sense?

Notes for Questions 4 and 5:
Caputring remarks on what the organisation has implemented to how this has helped develop the participant professionally. Probe for discussion.

The final three questions are relaxed, and are designed to allow the participant an opportunity to directly raise any questions or concerns with the research in the privacy of the interview.

5. How do you feel with regards to this research?

6. Do you have any questions or concerns regarding the research project?

7. *Do you have any other questions or things you wish to discuss?*

Notes for Questions 5, 6 and 7:
Questions 5 and 6 and 7 are a chance for the participant to ask any question they may have and to find out further information.

End of Interview checklist:

- Thank the participant for their time and insights.
- Reiterate the options on the consent form – in particular, double check if they wish to receive a transcript or the recording of the interview.
- Outline the next steps…
  - Diary Study conclusion
  - Transcriptions
  - Feedback
- Ensure participant has contact details (provide a business card if possible).

Appendix 8: Ada Lovelace, Babbage's Analytical Engine and Note G

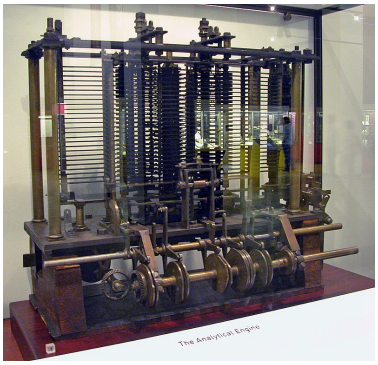


Figure 6.7: 1840 portrait of Ada Lovelace by Alfred E. Chalon and trial model of Babbage's Analytical Engine, Science Museum, London [Hollings et al., 2018].

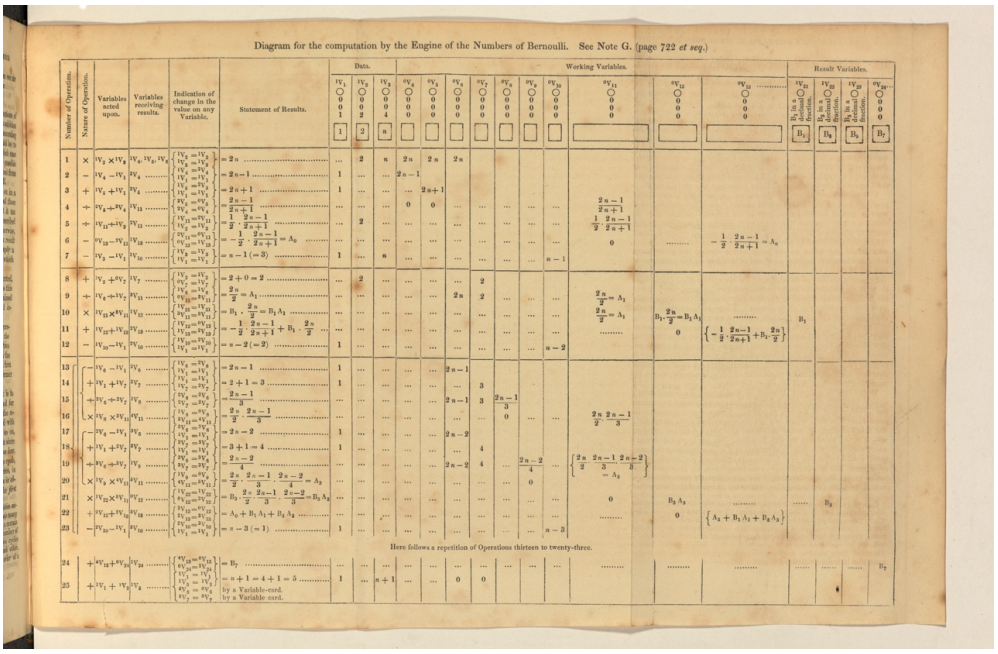


Figure 6.8: Note G, Lovelace's algorithm to compute Bernoulli Numbers using the Analytical Engine [Hollings et al., 2018].

Appendix 9:
Systematic Literature Review Bibliography

ID **Reference**

slr01 Gupta, V., Kapur, P., and Kumar, D. (2017). Modeling and measuring attributes influencing DevOps implementation in an enterprise using structural equation modeling. *Information and Software Technology*, 92:75–91

slr02 Dennehy, D. and 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

slr03 Fokaefs, M., Barna, C., and Litoiu, M. (2017). From DevOps to BizOps: Economic sustainability for scalable cloud applications. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 12(4):25

slr04 Ghezzi, C. (2017). Of software and change. *Journal of Software: Evolution and Process*, 29(9)

slr05 Wettinger, J., Breitenbücher, U., Falkenthal, M., and Leymann, F. (2017). Collaborative gathering and continuous delivery of DevOps solutions through repositories. *Computer Science-Research and Development*, 32(3-4):281–290

slr06 Chen, L. (2017). Continuous delivery: overcoming adoption challenges. *Journal of Systems and Software*, 128:72–86

slr07 Airaj, M. (2017). Enable cloud DevOps approach for industry and higher education. *Concurrency and Computation: Practice and Experience*, 29(5)

slr08 Fitzgerald, B. and Stol, K.-J. (2017). Continuous software engineering: A roadmap and agenda. *Journal of Systems and Software*, 123:176–189

slr09 Sebastian, I. M., Ross, J. W., Beath, C., Mocker, M., Moloney, K. G., and Fonstad, N. O. (2017). How big old companies navigate digital transformation. *MIS Quarterly Executive*, 16(3):197–213

slr10 Kneuper, R. (2017). Sixty years of software development life cycle models. *IEEE Annals of the History of Computing*, 39(3):41–54

slr11 Karl, H., Dräxler, S., Peuster, M., Galis, A., Bredel, M., Ramos, A., Martrat, J., Siddiqui, M. S., Van Rossem, S., Tavernier, W., et al. (2016). DevOps for network function virtualisation: an architectural approach. *Transactions on Emerging Telecommunications Technologies*, 27(9):1206–1215

slr12 Sun, D., Fu, M., Zhu, L., Li, G., and Lu, Q. (2016). Non-intrusive anomaly detection with streaming performance metrics and logs for DevOps in public clouds: a case study in aws. *IEEE Transactions on Emerging Topics in Computing*, 4(2):278–289

slr13 Takimoto, M., Komine, H., and Tamura, K. (2016). Network DevOps solution for creating new network services. *FUJITSU Sci. Tech. J*, 52(2):8–12

slr14 Wettinger, J., Breitenbücher, U., Kopp, O., and Leymann, F. (2016). Streamlining DevOps automation for cloud applications using TOSCA as standardized metamodel. *Future Generation Computer Systems*, 56:317–332

slr15 Oliveira, F., Eilam, T., Nagpurkar, P., Isci, C., Kalantar, M., Segmuller, W., and Snible, E. (2016). Delivering software with agility and quality in a cloud environment. *IBM Journal of Research and Development*, 60(2-3):10:1–10:11

slr16 Kim, M., Mohindra, A., Muthusamy, V., Ranchal, R., Salapura, V., Slominski, A., and Khalaf, R. (2016). Building scalable, secure, multi-tenant cloud services on ibm bluemix. *IBM Journal of Research and Development*, 60(2-3):8:1–8:12

slr17 Ranchal, R., Mohindra, A., Manweiler, J. G., and Bhargava, B. (2015). RADical strategies for engineering web-scale cloud solutions. *IEEE Cloud Computing*, 2(5):20–29

slr18 Sill, A. (2015). Emerging standards and organizational patterns in cloud computing. *IEEE Cloud Computing*, 2(4):72–76

slr19 McLarnon, B., Robinson, P., Milligan, P., and Sage, P. (2014). An iterative approach to trustable systems management automation and fault handling. *Journal of Network and Systems Management*, 22(3):366–395

slr20 Lapham, M. A. (2014). Software sustainment–now and future. *CrossTalk*, 27(1):33–36

slr21 Pass, S. and Ronen, B. (2014). Reducing the software value gap. *Communications of the ACM*, 57(5):80–87

slr22 Roche, J. (2013). Adopting DevOps practices in quality assurance. *Communications of the ACM*, 56(11):38–43

slr23 Feitelson, D. G., Frachtenberg, E., and Beck, K. L. (2013). Development and deployment at Facebook. *IEEE Internet Computing*, 17(4):8–17

slr24 Hosono, S. (2012). A DevOps framework to shorten delivery time for cloud applications. *International Journal of Computational Science and Engineering*, 7(4):329–344

slr25 Obstfeld, J., Knight, S., Kern, E., Wang, Q. S., Bryan, T., and Bourque, D. (2014). Virl: the virtual internet routing lab. In *Proceedings of the 2014 ACM conference on SIGCOMM*, pages 577–578. ACM

slr26 Bass, L., Weber, I., and Zhu, L. (2015). *DevOps: A Software Architect's Perspective*. Addison-Wesley Professional

slr27 Cois, C. A., Yankel, J., and Connell, A. (2014). Modern DevOps: Optimizing software development through effective system interactions. In *Professional Communication Conference (IPCC), 2014 IEEE International*, pages 1–7. IEEE

slr28 Császár, A., John, W., Kind, M., Meirosu, C., Pongrácz, G., Staessens, D., Takacs, A., and Westphal, F.-J. (2013). Unifying cloud and carrier network: Eu fp7 project unify. In *Utility and Cloud Computing (UCC), 2013 IEEE/ACM 6th International Conference on*, pages 452–457. IEEE

slr29 Smeds, J., Nybom, K., and Porres, I. (2015). DevOps: a definition and perceived adoption impediments. In *International Conference on Agile Software Development*, pages 166–177. Springer

slr30 Walls, M. (2013). *Building a DevOps Culture*. O'Reilly. Sebastopol, USA

slr31 Mohamed, S. I. (2015). DevOps shifting software engineering strategy Value based perspective. *ISOR Journal of Computing Engineering (ISOR-JCE)*, 17(2):51–57

slr32 Dyck, A., Penners, R., and Lichter, H. (2015). Towards definitions for release engineering and DevOps. In *Release Engineering (RELENG), 2015 IEEE/ACM 3rd International Workshop on*, pages 3–3. IEEE

slr33 Lwakatare, L. E., Kilamo, T., Karvonen, T., Sauvola, T., Heikkilä, V., Itkonen, J., Kuvaja, P., Mikkonen, T., Oivo, M., and Lassenius, C. (2019). Devops in practice: A multiple case study of five companies. *Information and Software Technology*, 114:217–230

slr34 Veres, O., Kunanets, N., Pasichnyk, V., Veretennikova, N., Korz, R., and Leheza, A. (2019). Development and operations - the modern paradigm of the work of it project teams. In *2019 IEEE 14th International Conference on Computer Sciences and Information Technologies (CSIT)*, volume 3, pages 103–106

slr35 Koilada, D. K. (2019). Business model innovation using modern devops. In *2019 IEEE Technology Engineering Management Conference (TEMSCON)*, pages 1–6

Appendix 10: Definition Response Themes

Definition One - Positive Themes

Accurate	Delivery	Productivity Enhancement
Agile	Empowered	Quality
Aspirational	Impactful	Responsibility
Automation	Information	Role
Broad	Methodology	Simple
Business Outcomes	Metrics	Streamlined
Clear	Mindset	Structure
Collaboration	Multi-Disciplinary Teams	Succinct
Concise	Non-Technical	Tools
Continuous Improvement	Operational	Work Flow
Culture	Pertinent	

Definition One - Negative Themes

No Accountability	No Continuous Delivery	No Ownership
Automation	No Culture	No Reducing Time
No Behaviour	Empowered	Silos
Business Outcomes	Limited	Streamlined
Buzzwordy	Management Focus	Technical
No Collaboration	Methodology	No Testing
Complex	Multi-Disciplinary Teams	Tools
Continuous Improvement	Operational	Vague

Definition Two - Positive Themes

Accurate	Evolution	Reducing Time
Breaking Silos	Goal	Resources
Bridge	Informative	Role
Business Outcomes	Innovative	Simple
Collaboration	IT Services	Skills
Communication	Measured	Support
Continuous Improvement	Non-Prescriptive	Team
Culture	Non-Technical	Technical
Delivery	Organisational Process	Technology
Efficiency	Prediction	

Definition Two - Negative Themes

No Accountability	No Continuous Delivery	No Ownership
Academic	Idiotic	No Process
Adversarial	IT Services	No Quality
Archaic	ITIL	No Streamlining
Assumptive	Limited	No Testing
Buzzwordy	Management Focus	No Tools
Change	Meaningless	Obfuscatory
Clunky	Metrics	Operational
Collaboration	Misrepresentative	Opinionated
Complex	No Automation	Short
Development	No Behaviour	Too Specific
Disjointed	No Business Value	Unfocused
Division	No Communication	Vague
Evolution	No Culture	Verbose
Frustrating	No Delivery	
Generic	No Empowerment	

Appendix 11: Specimen Theme Coding for Case Study

ID	P. Theme	S. Theme	T. Theme	Date	P	Summary	Raw Text
I33	Culture	Decision Making	Measure-ability	12/01/2016	P6	Culture shift already evident with Dev team	I've seen it change since I've been here. So, a couple of the developers have never used Jira, and now we're using it everyday to managed our workload. I can see that there is change at AF, definitely since I've been here... and how we work and how we get the business involved in every decision we make because it's going to save time and money.
I39	Job Crafting	Relationship Crafting	Culture	12/01/2016	P7	Dealing with resistance by trying to buy-in individuals to the DevOps agenda	I'm basically pushing through the agenda that I've got and feeling ahead to see where it meets resistance and trying to then break that resistance down individually.
I46	Process	Culture	Continuous Integration	12/01/2016	P6	DevOps from the tester - deployment and operations is the focus	I think its like a structure or guidelines of sorts. But DevOps is deployment and, something ... that stands out for me... how we deploy and I guess if I said Agile... is that a cop out? DevOps is an agile approach? Yes, DevOps, how we develop as Operations I guess, how the Developers all work together, how we deploy, how we redeploy and stuff. It's all Agile, at least my take on it.
I03	Legacy Systems	Business Management		25/01/2016	P10	DevOps will bring improved process, quality and greater accountability for the developer	I think there's a roadmap that Jon would have, and I think there's certain would like to have and nice to have features on AFI. But I think mostly that is either done or is being pushed into Harrier as a feature. So he would definitely have a bigger and more accurate view of where AFI is. But I think there is an understanding in the business that it is end of life.
D25	Job Crafting	Relationship Crafting	Automation	23/02/2016	P7	Reaching out to P13 and providing necessary training and support. Strategic element in the relationship crafting by leveraging P13's intellectual curiosity.	Last week <participant 13> went on a three day PowerShell course. Mainly this was for him to become more productive in his current job. We have told him that these skills are transferable to Azure Power Shell and could be very useful in helping us automate much of our environments. Next time <participant 3>, <participant 12>, <participant 13> and myself are in we will talk this through and see if there is any interest in <participant 13> doing Azure PowerShell work
D43	Responsibility	Release		04/04/2016	P1	Responsibilities in release procedure	So the actual release procedure worked really well on the 24th. Jon made a release bullet point list of about 12 things, Nitesha handled anything data migration wise. I took the website offline, pressed my gitflow button in source tree and the magic happened.
I89	Culture	Responsibility		03/05/2016	P8	Hostile response from Ops - subsequently put P8 off doing anything with Ops or infrastructure related tasks	The last time I tried to do anything on the operations side, or put my nose in I got shouted at. I just asked Operations about some upgrades and if they've upgraded one of the servers to the HTTPS and I got moaned at saying it's my responsibility, and then he goes, that for this I go to him, so what am I meant to do then?
D54	Job Crafting	Relationship Crafting	Task Crafting	11/05/2016	P1	New laptop, Ops error but P1 rectified and provided solution to Ops	New laptop is here and has 16GB RAM, i7 and SSD. Seems a lot faster so far. What was interesting was <P7> got OPs to install a list of software. All was as expected except for SQL Server. We wanted Express with Management Studio, but got just Management Studio. So I fixed that myself and gave OPs the correct exe to use for the rest of the team's laptops.

ID	P. Theme	T. Theme	T. Theme	Date	P	Summary	Raw Text
D65	Job Crafting	Cognitive Crafting		11/08/2016	P4	Reflecting that undertaking back end work was good for him	Getting the CSS working on the front end was a big achievement for me too. I like to get involved in all aspects of Harrier, so the back end work is another string in my bow.
D93	Ownership	Responsibility	Control	20/10/2016	P10	Key Person Reliance - One individual who is leaving owned a lot of the Azure work	So we are losing two of our team - one to relocation, the other to contracting lucrativeness. The former is the real concern, as he has by and large owned a lot of the Azure related work within the team.
D100	Job Crafting	Relationship Crafting	Release	03/11/2016	P4	P4 not only enjoying the work across the stack, but also considering how his contributions can make a positive difference for a department he has never worked with before.	I have completed the invoice feature I was working on previously. While this has yet to be released (although it is now in the pipeline for release), a demo has been given to the end users with favourable reactions and the feeling this will make things much easier for them. It was good to learn and apply the new technology previously mentioned in my diary to feature. While an enjoyable undertaking, it was challenging too as this was the first time I worked with the invoicing team, and mainly due to the differing terminology. The system they presently use is a bit chaotic, I believe Harrier will significantly improve things for them.
I118	Release	Process		06/12/2016	P2	Dev team handling releases of Harrier now	I think basically the software team are doing the releases generally with Harrier and, I think, the Ops team, I'm not quite sure what they do, but I think they do a little bit of SQL and stuff to help out, and that sort of thing, so it's a bit of cross over on both parts.
D115	Culture	Decision Making	Business Management	15/12/2016	P10	Application of DevOps principles - Development and IT Support to work closer - most challenging, culturally.	We host on IIS (Microsoft's webserver product). This will require Support and Development to work closely together to monitor and maintain the environment. Culturally, this would be the most challenging path to take, but may provoke the most change in how the two sides currently work. (Or it could devolve into a living nightmare of passing the blame and fence building, but hey, best to be positive in our outlook.)
D124	Business Management	Process		13/01/2017	P11	Migrating users to Harrier from AFI not well handled	Encouragement of use of software has not been handled great, would have been good to see some floor walking as such of support to just help with issues as they happened. This would help to build a better rapport with users and support.
I212	Knowledge Management	Transformation		16/02/2017	P8	Team-level transformation in knowledge management where new members can access easily existing knowledge.	Personally I think the skill set of the whole team's gone up and that's mainly through the feedback via call requests and this gives the opportunity to share some skills and knowledge and techniques between say new employees and existing ones who haven't been on the system before. So I think that's very, very positive.
I154	Culture			20/03/2017	P2	Silos - but not necessary 'silos' - they do work together	There are definitely still two very distinct departments, but yeah, I think we work well together. I wouldn't necessarily say it's a problem.
I172	Culture			20/03/2017	P5	DevOps culture - self-sufficient team - blending of functions	DevOps is where a team of developers become self-sufficient in terms of their IT operation. It's a blending, a melding of the typical operations skills with the software development skills, certainly in Anglia Farmers, with a view to making us self-sufficient and more efficient.

Appendix 12: Case Study Theme and Quote Index

Case Study Time Period A – Diary Index

ID	Themes	Raw Text
D05	Job Crafting	The agenda is to communicate the developer architecture vision to operations and what tools and processes are needed to make sure this works on Azure.
D12	Ownership	The upshot is that "Dev" will lead all the deployment and configuration work except where "Ops" are needed to make changes that Dev do not have access to (e.g. DNS settings).
D17	Job Crafting	Ops are not proactively looking to get involved in the Harrier rollout.
D19	Legacy Systems; Continuous Integration	The contrast is marked - no CI, few unit tests - and shows how important getting that build pipeline up and running really is. Thinking about environments and how to deploy code quickly to them is something that needs to happen right at the start of the development process.
D25	Job Crafting; Automation	Last week <name omitted> went on a three day PowerShell course. Mainly this was for him to become more productive in his current job. We have told him that these skills are transferable to Azure PowerShell and could be very useful in helping us automate much of our environments. Next time <name omitted>, <name omitted>, <name omitted> and myself are in we will talk this through and see if there is any interest in <name omitted> doing Azure PowerShell work
D28	Job Crafting	He gave a really interesting talk, some of which really resonated with me regarding the situation at AF. Most interesting was his view that commitment from senior management is essential for the success of creating a DevOps working environment. Without that commitment, no matter how proactive the development team is, the barriers between dev and ops are not going to come down on their own.
D31	Job Crafting; Responsibility	I do feel if OPs were more helpful on the Azure side we would be nearly a sprint ahead by now. I think the mistake we made was doing Azure ourselves. We are now seen as able to do it for now, but keep having to do more and more. A better approach would have been to have had early requirements supported by management on the OPs team. I think our technical intrigue as developers has actually hurt us here.
D38	Job Crafting; Continuous Integration	I made all the azure web apps, Azure Power Shell runbooks, added a config transform to each microservice, added projects to Jenkins, Hipchat rooms, I think that's the big stuff. No OPs involvement was required.
D39	Legacy Systems; Decision Making; Measurability / Metrics	Optimistically, I had hoped that this would take an extra week to deliver but there was far more to it than I had anticipated (i.e. there was far more going on in AFI that needed to be replicated on Harrier than I assumed) and it has ended up being an extra 3 weeks in total.
D41	Job Crafting	After doing some reading up on Azure Stack (brings Azure cloud technology and benefits to on premises) I decided to run this past the senior developers, <name omitted> and the Ops team. I set up a meeting in a room with a TV and we watched a couple of Microsoft-produced videos on Azure Stack that were mainly non-technical in nature.
D43	Responsibility; Release	So the actual release procedure worked really well on the 24th. <name omitted> made a release bullet point list of about 12 things, <name omitted> handled anything data migration wise. I took the website offline, pressed my Gitflow button in source tree and the magic happened.
D54	Job Crafting	New laptop is here and has 16GB RAM, i7 and SSD. Seems a lot faster so far. What was interesting was <name omitted> got OPs to install a list of software. All was as expected except for SQL Server. We wanted Express with Management Studio,but got just Management Studio. So I fixed that myself and gave OPs the correct exe to use for the rest of the teams laptops.

Case Study Time Period A – Interview Index

ID	Themes	Raw Text
I02	Culture; Legacy Systems; Quality	Certainly, when certain people were working on AFI predominately, there was a bit of resentment ... like I'm not actually on the new project, and everyone else is getting to do this new, exciting stuff and they're stuck doing all this legacy Visual Basic (VB) code, which no one really likes. The main problem with it is there is no separation of concerns... you can't pull one part out and replace it with another. You can't do incremental changes, so if you pull one part out... it's like tugging on threads, and it all starts to unravel"
I03	Legacy Systems	But I think there is an understanding in the business that it is end of life.
I05	Decision Making; Quality; Legacy Systems	The lack of architecture meant you couldn't tease it apart and there were no layers... so you couldn't take this layer out and replace it, or I'll take this thing out and there would be a nice clean interface here that I could implement differently. So, it was a bit of a mess, and the decision was taken to re-write it.
I06	Responsibility; Culture; Process	Blurring the lines between what constitutes development work and ongoing support, deployment and management of the real estate. Responsibilities will merge and become everyone's responsibility. Always had comfort from the fact that there's a certain point you're not responsible for your work anymore. If you don't live with your mistakes as a developer, you don't really improve as a developer.
I10	Process; Culture	I don't see them being involved in the actual sprint which is developer focused. But I could see them being involved in the release.
I19	Culture; Responsibility; Resistance	The whole Azure thing, the whole third party who used to manage the servers. I think there's a lot of politics there too, that holds stuff up. It will be slow because there will be resistance from different managers and people who won't necessarily make the decision. Because they know the people in those companies, and you're much more likely to do business with a friend, than do it a new way. The 'I've been using him for 10 years' mentality.
I20	Legacy Systems; Process; Quality	I think AFI has been kept out of that. It seems (AFI), very... well I wouldn't even call it Waterfall, rather a 'do it as it comes' very reactive, I don't know what the word is for that to be honest. They're not doing it in an Agile way.
I23	Work Identity; Legacy Systems	I remember when I first started it was sort of do this, deploy it and hope it works. Yeah, hope for the best! It really was like that.
I30	Process; Culture; Continuous Integration	DevOps, how we develop as Operations I guess, how the Developers all work together, how we deploy, how we redeploy and stuff. It's all Agile, at least my take on it.
I31	Process; Responsibility	I think IT support, maybe a year ago, would have been split Hardware / Software. But I think now, they're mainly moving, shifting towards the Hardware. When the issues get raised they run all the systems and do what they want. Obviously, they'll look at the ticket and then if they can fix it, they'll fix it. If they can't, they assign it to our team.
I32	Culture; Process;	Anybody who's come from an old school approach to developing software might not embrace it initially.
I33	Culture; Decision Making; Measurability / Metrics	I can see that there is change at AF, definitely since I've been here... and how we work and how we get the business involved in every decision we make because it's going to save time and money.

I34	Culture; Resistance; Work Identity	A developer's never going to go and install a monitor for someone in the business, they (Ops) will always do that.
I35	Work Identity; Automation; Continuous Integration	Just going to make my CV better aren't they, surely? Unless they build or get a robot to completely do my job and completely automate everything. I'm going to learn from it, and I think they need a tester. As good as Developers' code may be, there's always going to be integration and look and feel issues you know. So, it's only going to improve my skills.
I36	Quality; Measurability / Metrics	Business will be able to work quicker; they won't have system issues. They should be able to process more orders, things over the phone because the system will be better, they'll be able to get more work done in their working day, so it's certainly going to mean that we (AF) can take more business.
I37	Job Crafting	With the size of the team we've got now, there is a place for a full time business analyst, and I've tried to argue that one. I've won the argument, but it's never transpired and hence one of the reasons I'm doing so much business analysis as it needs doing.
I38	Culture; Resistance	So, there's a view of 'ours and theirs' and 'theirs and ours. Then there's the whole politics of the third party infrastructure management company: how they fit in; what's their view of what we're trying to do; what their view is of working in the cloud.
I39	Job Crafting; Culture	Pushing through the agenda and feeling ahead to see where it meets resistance and trying to then break that resistance down individually.
I41	Responsibility; Automation; Culture	I'm looking for automation down the pipeline, so I'm expecting the responsibility of the two teams will be to keep this automated pipeline running all the time with a fairly equal responsibility but obviously with an emphasis on Dev not to introduce crappy code that breaks it, and Ops to not fiddle with security settings without thinking it through.
I42	Culture; Transformation	A lot of it is not really DevOps, in that we're producing a much better system than we have currently, but the ability to deliver that system and keep it running is a big thing.
I43	Culture; Resistance	There is no outright opposition, it's more just inertia due to their own observations of a 'default position of sit tight because this has always worked, even though it's a bit messy.
I44	Job Crafting	Slowly winning people round to this new way of doing it.
I45	Legacy Systems; Knowledge Management	Yes, as I'm learning the new technologies, I'm having to put myself into 'learning mode' and then, for example, when something has gone wrong with AFI or something hasn't gone right in testing, I have to then, take that cap off then try and get my head back into the other mode, and the swapping just takes a little bit of time. Obviously, when you're learning, things go out of your head and when you come back, you're like, well what was I actually trying to do and that's the hardest part, it really is when you're trying to learn. If I knew it all, it wouldn't be too bad, but learning it and swapping about is difficult.
I46	Business Management	It's all to do with money and saving numbers... that's what I believe it is. If they can save money on support or programmers by doing something, they will.
I49	Process; Culture	We sometimes involve them at the starting point of any project for what would be the project requirements in terms of the technologies and hardware and everything.
I51	Culture	Would like to see it happen. There's not very many people who will actually do both.
I53	Culture; Business Management	I might be talking to my colleague; they might be listening... or they might be talking and you hear what they're doing... and you go that's not going to work straight away. You can hear what they're actually saying and vice-versa.
I65	Culture; Responsibility	My colleague, he's a bit more old school, so he might take an approach different to say <name omitted>, who has these new ideas. Or it could be that development do not fully

		understand what Operations is doing and vice-versa. We don't fully understand each other's roles yet and there has never been any full clear definition on who is responsible for or should take ownership of what bits.
I67	Ownership; Culture	In house, we maintain it, we look after it, if anything goes wrong, it's our fault and we protect it and do anything with it. Development are very eager to get cloud bits and bobs going and they're saying we'll pay the money, we'll get Microsoft to sort it out for us <name omitted> has been looking after them for the past 5 years anyway, they're his baby, and now they want to throw them out of the window and go, we'll get a new baby. I think it's more about the protecting of his server and he wants to still be able to maintain it himself, than for us lot to sit there in the corners loose limbed and pay Microsoft. They are his pets, that's how I'd perceive it. A server is a server, once you start naming them, then you get sentimental.
I69	Culture; Resistance	Our managers and the managers of them maybe don't respect or understand what we talk about all the time. That's another barrier above us, which can be a conflict. So that culture has to change there at some point, and it's not about if it has to change, but it has got to change.
I78	Responsibility; Process	Never heard of the term until recently... last 12 months. To me it means the link between writing some software and how its going to be deployed on the system... and who takes responsibility. So it's really trying to define the roles.
I79	Knowledge Management; Responsibility; Process	Been driven more from our development team. Historically, AF has employed third party developers, it's the first time we've done a big project with in-house development and the team that we obviously have. So I guess it's a learning curve for senior management in the business.
I80	Responsibility; Decision Making; Business Management	There are still silos of Dev and Ops. I think... short of bringing someone in a DevOps role who bridges both parts, which potentially could cause more problems as you bring three people to the table. At least with two people you can kind of knock their heads together and agree that sits there and that sits there... which sometimes is what it's almost felt like
I81	Job Crafting; Resistance	Two main characters who don't always see eye to eye. I have to listen to <name omitted> from a support point of view and knowing there are some things he can setup that <name omitted> isn't 100% aware of. There are some things from <name omitted> from a development point of view that's in his language, and it's almost like I'm sat in the middle as a layman, I read it as this without trying to bring any technical terms to it.

Case Study Time Period B – Diary Index

ID	Themes	Raw Text
D58	Responsibility	<Name omitted> and <name omitted> are now trained on Harrier and are starting to take support calls.
D60	Job Crafting	I am guessing Microsoft updated the portal overnight and we didn't have something required in that version. Anyway the big question is why did I handle this? It's a virtual machine (VM). There are no scripts here and I was using a user interface with the Microsoft guy. Nothing about being a developer helped here. I fixed it because I want our nice front-end CI running again. We should reassert our push with higher management to be handing management of the Operating Systems (OS) and Azure to Ops.
D64	Culture	I think there is now much less communication between Dev and Ops following <name omitted> and myself having to move desks due to lack of space, given the appointment of a BA and additional Software Developer.

		Our weekly meetings with <name omitted> have also ceased, and at present, have not resumed. <name omitted> also stated to me that he does not feel it relevant for him to sit in meetings with us and <name omitted>, where we discuss Ops specific and facility tasks. While I agree that some development tasks are not relevant to us, I feel we do need to know if anything will affect the network, AFI or user experience.
D65	Job Crafting	Getting the Cascading Style Sheet (CSS) working on the front end was a big achievement for me too. I like to get involved in all aspects of Harrier, so the back end work is another string in my bow.
D67	Legacy Systems	I am now working more on Harrier than AFI. It has been particularly good to apply my skills with XML to Harrier too. The overall workload on AFI seems to have drastically reduced. In turn, I feel much happier to be working on Harrier than AFI now.
D69	Work Identity	My focus in the intervening time has been almost exclusively on software development. The two largest features of AFI - ordering and invoicing are in the process of being implemented in Harrier. The time I would have spent giving consideration to Ops issues, is instead being used to work with our BA. In a way this is a more traditional take on being a software developer, and it now feels a bit odd.
D70	Work Identity	<Name omitted> has largely taken over the Ops side of things, though I feel a sense of frustration at having to step away from a number of open issues. I think this highlights the extra demand on time that DevOps places on a developer. At present, I don't feel I can devote time to everything and still deliver on the development side of things.
D76	Job Crafting	<Name omitted> and <name omitted> from the development team have been giving assistance to myself and <name omitted> on how to start and stop the Azure reference site. This is essentially the version of Harrier which is used for both testing and demonstration purposes. We can control this through commands, but in particular through the Hipchat tool used by the development team.
D77	Job Crafting	Additionally, we are socialising well with the development team, and are looking forward to attending a crazy golf team building day with them and others in the business.
D81	Decision Making; Accountability	It is tricky to get time with <name omitted>, as they are also so busy supporting the business. <name omitted> being our proxy, the key stakeholder in how AF is wanting the new system to be built does cause delay in readying work in time for Dev sprints. This of course will also cause significant delay to release.
D82	Work Identity; Job Crafting	On reflection though, I am definitely getting to learn, play with and apply new technologies as part of the overall delivery objective of Harrier. I still have a huge desire to continue learning too. Also, as I have generally always been a front-end developer this is new, given its back-end functionality, as such, I have been writing more C#.
D83	Job Crafting	I've been working on an invoice pdf converter for Harrier. This essentially involves the conversion of Extensible Markup Language (XML) into a pdf invoice. Again, this is very new to me and the first time I've ever looked into such functionality. Nonetheless, it is great fun and has led me to investigating looking at FO.net(a C# library) as a possible avenue to developing a solution.
D84	Work Identity; Transformation	I am starting to find that the feature stories I work on are involving elements of both front and back end work. Subsequently, I no longer see myself as a front end developer, but rather a full stack developer, and I believe this makes me a much better developer.

D85	Job Crafting; Transformation	I also enjoy being able to move between both and I believe this benefits the business too, that operating in a full stack manner is more efficient. I also like the change too, if I did purely front-end for instance, I would probably end up getting bored.
D88	Knowledge Management;	I've taken a job with Muddy Boots Software. They have 3 creaky codebases and want to bring in someone to oversee bringing DevOps and CI into their organisation, alongside a more micro-services type architecture. They've basically made a role for me, which is awesome.
D90	Collaboration; Job Crafting	I suggested to <name omitted> that a Harrier Implementation Group be set up to manage the roll-out of Harrier. We have our first meeting later this week. The group's responsibilities include User Acceptance Testing (UAT), training and platform - the last two of which normally are the responsibility of <name omitted> and <name omitted> respectively.
D92	Job Crafting	There is still no further progress on a hosting decision as <name omitted> has not yet arranged for InTouch to come in and talk through our options. My feeling is that the only sensible option would be on-premise while our Internet connection is anything but bullet-proof. Surprisingly, at my last meeting with him, the CEO seemed to be encouraging us to look at the cloud option - I think the 'serviced platform' idea is appealing.
D93	Ownership; Responsibility; Control	So we are losing two of our team - one to relocation, the other to contracting lucrativeness. The former is the real concern, as he has by and large owned a lot of the Azure related work within the team.
D94	Knowledge Management; Ownership; Business Management	I guess this highlights how easy it is to rely on one person to get certain tasks done; when you are in full-flowing Dev mode, you don't stop to consider how certain things are getting achieved, just that they are getting achieved. So, this poses a bit of a problem for us in that we need to cover the impending skills loss. Ideally we will spread the responsibility across the team this time, and avoid a future repeat of this situation. But, alas, I fear this dance is performed in many development teams, over and over again.
D95	Knowledge Management; Release; Responsibility	My focus this last week had been about handing over as much knowledge as possible. To help with this <name omitted> allowed me to bring a handful of tickets into this and next sprint that I know will be particularly difficult or stretching in Ember terms. Other focus is on passing across some Azure and Jenkins experience.

Case Study Time Period B – Interview Index

ID	Themes	Raw Text
I73	Process; Quality; Business Management	The Harrier project is better because we embrace DevOps, and you know, we think about it as developers, and it makes our software a lot easier to write and you know, our releases have so far been a lot cleaner. So it's definitely improved things. I'm not sure that they're sold on the value of it in the same way that the developers are. I don't know why, but I think <name omitted> takes the lead on it really, rather than anybody above.
I74	Legacy Systems; Process; Quality	Getting everybody thinking in a much cleaner mindset, you know they're used to doing quite dirty hacks in AFI. So, getting them thinking about this is a clean project, we'll do it in a clean way... that kind of thing.
I77	Legacy Systems; Process	We have to communicate data from Harrier back to AFI, so there's that side of things and having to get it back the whole time. So, the single point of orders is within AFI. We call it the AFI RESTful service, and it sits here. It has some APIs that Harrier can hit, and it updates the AFI database.

I88	Release; Process	We're waiting on stuff... we've waited for decisions and to have various sign off meetings. We can't progress further until things are signed off from higher up.
I89	Culture; Responsibility	The last time I tried to do anything on the Operations side, or put my nose in I got shouted at. I just asked Operations about some upgrades and if they've upgraded one of the servers to HTTPS, and I got moaned at saying it's my responsibility, and then he goes, that for this I go to him, so what am I meant to do then?
I90	Legacy Systems	With Harrier, Ops hasn't been involved too much, which I think has been a good and bad thing. I guess it's not working towards DevOps, but if there's nothing to fix, they are kind of doing it on their own stead. Realistically, to us, they are controlling it, and we don't have much input. To my eyes, it looks good, it performs well and from what I can see, the users are happy with it. AFI is 10 years old, so it has its own things to do what you can do. Harrier has much more new things which you can do things in a much quicker and nicer way. It's just not worth spending as much time on AFI. It is frustrating, because you know you can do things better, but there is the case on quality and time periods.
I93	Responsibility; Business Management	Senior management need to specify the principle responsibilities of those different groups. It's all well and good saying its DevOps and it's combined, but there's two separate teams there who do things in different ways. So I think management's job is to specify where the dividing line is, even though with DevOps there's not supposed to be a dividing line.
I94	Decision Making; Culture	They're broadly on board with the whole Azure platform, they're looking at doing this on-premises version which hasn't been released yet which Microsoft are looking to release later this year.
I100	Culture; Process	On the one hand, if Ops have the attitude that we don't want to get involved, then it kind of makes it easier for us so long as senior management say well fine, they're not getting involved, then Dev can do what they want to do, and you can't object to it. It makes our life easier in some respects, as we get to pick and choose the things we want to do in terms of tools, techniques, processes and stuff. But, I think as they're such an integral part of the company when it comes to fielding user queries and those kind of problems and things like that and the general day-to-day running of the office, they've got to be onboard, certainly with releases and what's going out. They need to know where to look through logs and things like that so they can relay better information to us. If they're going to be a first port of call to users coming in, if they know where the logs are, what the services are and what features are affected, they can say this things come in, here's the relevant log entries, just as a basic example.
I101	Process; Culture; Resistance	We've had nothing to actually support or a conversation or document to say this is what we've done, we're handing that over to you. If you haven't handed it to us, how are we supposed to deal with it?! You carry on and support it, until you finish it or send us exactly what you want us to support.
I102	Process; Legacy Systems	Harrier is well from a build and needs to be done by development and then handed over.
I103	Culture; Control	It's very similar to running a server, very much the same principle but you go about it in a different way. I think it sounds a good thing. Because it's all lumped into one. It's one interface where you can do everything in one lump rather than fishing yourself around the server or creating the roles, where they are already there. And to me, looking at it, I would definitely have Azure running the same interface with us. But I'm more prone to having it internally than externally.

I104	Legacy Systems; Process	There's ways round and he has to find what users can't do. Harrier, I'm hoping, will eliminate that. So to a certain extent, my theory is that Harrier will make <name omitted> be able to do other things, rather than faffing around with AFI/Harrier. Call it Harrier, whatever you like. Harrier will hopefully replace the problems you have to deal with AFI. Then he would have time to do other things.
I105	Business Management; Culture; Resistance	The only reason we hear about DevOps is through <name omitted>, but he doesn't manage Operations. I feel at the moment, <name omitted>'s got one size shoe that he wants DevOps to fit, and we're not Cinderella. I feel like in his mind, he knows what he wants for DevOps, but that might be different to how we see it at the moment. I feel we're not communicating enough to get any vision across. Although I wouldn't want to class us as ugly sisters... but yes. Regrettably, at the moment, I don't feel DevOps has moved as far forward as I would have liked it to. There's definitely been approaches towards it, but myself and <name omitted> are involved in development meetings. We've tried to involve them (Dev) in some of our bits as well, but it seems to be at the moment the idea of <name omitted>'s idea of DevOps to what we would like it to be is slightly different. Our manager isn't moderating that, so it's almost like a free for all. I think senior management and above, including the CEO... I think their role should not be just to moderate it, but to show by example. If they don't understand it or show interest, it will never motivate us to look at it. I think the reason it hasn't gone as well as we'd like it, is that both Dev and Ops should report to the same manager.
I108	Job Crafting; Work Identity	I've been on a power shell training course. On the three day course I learned power shell. An interesting fact as I went to that to learn about active directory, exchange and group policy. But I think the impression for DevOps is that I'd be able to use that skill for Azure as well. So I think there was a bit of miscommunication there. I think <name omitted> expected me to come back and use power shell straight away for Azure. But the three day course didn't even touch on Azure. I've now got a book, with a big bit at the back of it, which is full of Azure. So, as I see more of this within Azure, it puts me off a bit where I see the simple couple of commands for active directory for reactivating an account or changing a list of active users or active computers running on the W32, then that's the thing I'm interested in. If it opens up in the future, I wouldn't mind delving into it. As a person, I've always wanted to learn more. But for my professional need, I feel I don't need that at the moment.

Case Study Time Period C – Diary Index

ID	Themes	Raw Text
D98	Legacy Systems	I did have to look at the RESTful web service for AFI in order to investigate why a few things were not working. In the end, there was an issue involving the wrong environment being used and issues around usernames and such. Thankfully, this was a relatively easy fix and did not interrupt my Harrier work. Otherwise, there has been no other AFI work.
D99	Legacy Systems; Process	Sadly, I had to do a couple of AFI tasks, but thankfully these were small and did not interfere much with my Harrier work.
D100	Job Crafting; Release	While an enjoyable undertaking, it was challenging too, as this was the first time I worked with the invoicing team, and mainly due to the differing terminology. The system they presently use is a bit chaotic, I believe Harrier will significantly improve things for them.
D103	Job Crafting; Culture	On a more personal note, I was very sad to hear that both <name omitted> and <name omitted> were leaving. I valued the relationship and friendship I had with <name omitted>

		especially, and as a group we have had many social nights out which I'm sure everyone will miss. I look forward to attending the Developer leaver's meal.
D108	Work Identity; Transformation	After learning a lot about the Ops side of the department from the handover with <name omitted>, I'm back on to primarily development, but handling the releases and candidate cutting as <name omitted> used to do. My role has changed so I that I can fill the role that <name omitted> left behind. I have more responsibility and am much more involved with Ops.
D111	Job Crafting; Quality	One gripe I do have is that I wish the way the UI is coded, in that it needs improving. This is something I have spoken to <name omitted> about, and I feel I have skill with UIs. I think he is onboard with the idea. In particular, I feel we need to code the UI to cater for multiple screen sizes.
D113	Job Crafting; Quality	I feel I am still improving my skills across the stack and am feeling positive about this. I maintain my focus on producing quality software and continue to be thorough in my approaches. On reflection, I like to think that what I produce makes others' lives easier. Additionally, when you see people using your software, and appreciate it, it feels good. This extends to other developers too, because good quality code is far easier to pick up.
D114	Culture	Unfortunately a conversation with one member of Support was rudely interrupted by the other. I finished the conversation and walked off. Having previously worked in Support, the customer was always more important than current tasks at hand. Quite upset by this, however the Support member I was talking to did pop to my desk to complete the conversation. Approachability of Support is very important, developing good reputation and confidence with customers they are supporting can go a long way to making an efficient workplace.
D115	Culture; Decision Making; Business Management	I foresee us remaining on in-house hardware for a number of years, as the subsequent costs and effort of moving will always be weighed up in light of other business development needs (when you have a working platform, feature development will always take priority.
D116	Culture	This will require Support and Development to work closely together to monitor and maintain the environment. Culturally, this would be the most challenging path to take, but may provoke the most change in how the two sides currently work. Or it could devolve into a living nightmare of passing the blame and fence building, but hey, best to be positive in our outlook.
D124	Business Management; Process	Encouragement of use of software has not been handled great, would have been good to see some floor walking as such of support to just help with issues as they happened. This would help to build a better rapport with users and Support.
D133	Job Crafting; Culture	Surprisingly the support member then comes back and provides further training to me on the file stream application. I suspect because I was pleased when I achieved what I needed to achieve on the initial request, it might have inspired them to want to help more. Later that day, the second member of support shares access to the Harrier training guide with this support member which is a surprise as they have not taken any interest in Harrier up until this point.
D150	Ownership; Transformation	We then looked into Azure Stack which seemed that we could continue with the DevOps model that we had already developed (i.e. DevOps within the Development team with Support overseeing security). Unfortunately, this was not to be due to the delay in the Azure Stack roll out. Instead, some new hardware is on order to host Harrier internally. This more or less puts us back to where we were with AFI in terms of DevOps responsibilities between Development and Operations.

D157	Job Crafting; Transformation	For me personally, it feels great that my development work column is currently clear and helping with others. On reflection, I feel I have come a long way since starting with AF; before I was strictly user interface (UI), but now like working with new technologies and working on different things. I also feel empowered to put forward my own ideas.
D160	Job Crafting	As more and more people in the company are starting to use Harrier we're seeing more live bugs appear, a minimal amount, but there has been some. On two occasions I've interacted with people outside of my team to get more information about it to do some debugging at their PC where it's happening. I expect this to happen more and more, which is a good thing rather than hearing it second-hand not seeing it happen for yourself.

Case Study Time Period C – Interview Index

ID	Themes	Raw Text
I110	Process; Responsibility	It's ended up with the development team, taking on Ops' responsibility rather than Ops getting involved more in Dev, but we do include them in release notes, and things like that. We give them visibility of what's coming up, so they should know what's coming down the pipe for releases.
I111	Job Crafting; Responsibility	We've taken all responsibility for setting up our own integration environments, testing environments, like provisioning stuff in the cloud for that, and basically having more of an eye on how our whole system hangs together and how you can replicate all those parts somewhere else if we need another environment.
I113	Culture; Business Management	They have to give us the space to try it, the approaches that DevOps entails. They have to accept that sometimes we're gonna fail, because this is new to us.
I115	Responsibility; Transformation; Process	I think people are more willing to get involved in kind of fixing problems wherever they happen to arise, so people aren't like, 'Ooh, I don't touch that bit of the system' or, 'I don't deal with the Azure bit'. Everyone kind of feels quite happy to take responsibility for various bits of it.
I117	Job Crafting; Transformation	I'm still not a front end developer, I'm still not an expert at infrastructure, but I kind of take that view of all of it. I feel it's my responsibility to at least understand what the impact is at those stages and what the trade-offs are for accommodating those bits of the system.
I118	Release; Process	The software team are doing the releases generally with Harrier.
I120	Culture; Business Management	It means bringing together the two disciplines of Development and IT Ops, making them work closer together hopefully to get economies of scale, insight and cultural uniformity so there's more cooperation and collaboration.
I122	Culture; Process	Look after their own destiny, they have their own capabilities to build, release, manage their environments, make their kit work and make sure they've got an environment that does what they need it to. I think a lot of this is about Dev taking on the Operations for their own environments.
I123	Job Crafting; Culture	I do have to build test data, test environments, but more I'm specifying to the Developers to help me build my environment. I'm certainly not a Developer with development skills. I have programmed in the past, I can do it, but they're quicker, better and I don't want to go and mess up something by building something not as good as they could do.
I125	Legacy Systems; Quality	When we develop something on AFI, as long as it works, it doesn't matter how it's done.

I127	Process; Transformation; Culture	So I think it's been exclusively Developer-led, and the involvement of Operations has been fairly small. We've made an effort but it hasn't particularly been seized upon, and I think <name omitted>'s got other priorities so there was no real forcing of the issue, so it's just naturally flowed in a very Developer-led way.
I129	Business Management; Process; Culture	Outside of <name omitted>, DevOps means nothing to any senior manager. I think <name omitted>'s got a lot going on, which so long as software is being produced and released, I don't think that the efficiency of it is high up on their priorities.
I133	Job Crafting;	I've never worked and I'm not that much expert in the front end, so it's new learning same time is challenging and also sometimes it's like … need help kind of thing. But yeah, it's exciting.
I134	Culture	Two separate departments, not working together as such.
I137	Legacy Systems; Process	There's certain departments who are using Harrier and other people are still using AFI to do the exact same thing, which, I know that is moving over slowly but surely. You have a differential between working with AFI and then when you were explaining it, only to find out they're not actually using AFI, they're using Harrier. If you didn't have to deal with AFI, in theory, it should be easier because we're going up the ladder with Harrier. AFI is out of date, so in theory you may not get as many problems with Harrier as you would with AFI. So if it wasn't there I would say that probably there would be less queries.
I139	Culture; Business Management; Process	I think the communication between Development and Operations has improved slightly. In some ways Operations moving a little bit further away from Development has given us almost an out of the box view of it, and allowed discussion between both Operations and Development to be a little bit smoother.
I142	Job Crafting; Business Management	I've learnt more business knowledge through understanding how Development work in their team. I understand DevOps is meant to help prevent conflict between two big teams like this and understanding and appreciating their views and concerns compared to our views and concerns, and seeing where there's a compromise with that.
I143	Culture; Business Managemet	Senior management should have played a bigger part. Because like any manager the first responsibility is to their own team so first of all they're always going to look to see how can they improve their team's efficiency and how it would benefit them. So I feel naturally that would always play bias towards whoever, even if the champion was in Ops' team, it'd be the complete opposite. The Ops champion would always favour their team, clearly.
I145	Work Identity; Transformation	I always understood my job role was, 'I'm Operations, they're Development, that's a clear-cut line'. Where now I understand what DevOps is, you kind of see how actually both are kind of intertwined together. And it obviously depends how much you deal with them, so I'm a bit more open minded than I used to be.
I146	Legacy Systems; Job Crafting	I think AFI, because of how much firefighting you have to do, can take up quite a bit of time. You're patching a sinking ship when a new ship's being built, so you think, 'Well, what's the point?
I147	Work Identity; Transformation; Culture	DevOps is the bit that some Devs like to do and some Devs don't. If people like to do it then they enjoy that grey line between operations and development, and enjoy setting up servers, scripts and all the kind of things that are somewhere in the middle. I'm a Devopeler, a Dev who does DevOps so sure, yeah. Whereas it's become apparent that some Devs don't want to do DevOps, and like just avoid it as much as they can at least from a Dev track.

I152	Decision Making; Measurability / Metrics	I think Azure Stack is going to be delayed until next summer, so it's not going to meet the time frames for phase 2 for us, so it's not an option any more.
I154	Culture	There are definitely still two very distinct departments, but yeah, I think we work well together.
I162	Culture	We've achieved it by I guess bringing two separate roles more closer in terms of the way that we've gone about the Harrier project. From earlier days they viewed life completely separately and I believe now they are much more joined up in thinking.
I163	Culture; Business Management	I think there's been a change in mindset, in working on an inclusive basis rather than an exclusive basis. I'm actually quite impressed how mature they've all been. I've not had to bang heads together, I've just had to sort of say, 'This will only work if you guys can make it work' and I think they've realised that themselves that 'it's going to cause me problems and if it causes me problems... Well actually if we just talk.
I166	Process; Quality; Measurability / Metrics	It was built to process invoices that were correct, not to process every invoice whether it was correct or wrong and that's the subtle difference I guess. I envisage in a year's time that anybody who retires or decides to leave in the invoice office we won't be replacing and it will be a key driver for the business in terms of keeping costs of the operation down.
I170	Culture	I would say it's pretty fully joined up. It's thought through. I would use the word collaborative. There are still strong characters. I don't so much think that an intermediate is required. Whilst they're strong characters they've learned how to channel their views and actually both see the end goals.
I171	Legacy Systems; Release	Well we're still reliant on AFI at the moment. It still is, as far as I'm concerned, the point of truth. Obviously when we release phase two, Harrier becomes the point of truth.
I172	Culture	DevOps is where a team of developers become self-sufficient in terms of their IT operation. A blending, a melding of the typical operations skills with the software development skills, certainly in Anglia Farmers, with a view to making us self-sufficient and more efficient.
I177	Job Crafting; Work Identity	I get into a role and I start to expand out to areas where I feel competent. I'll certainly offer anything I've got and part of that just happens to be an awareness of how things are done elsewhere.
I186	Work Identity; Transformation	I used to be predominantly working on front-end features and slowly moving on to back-end features.
I190	Legacy Systems; Quality	When the internet goes down at the minute, we would suffer. With the legacy system, it will still chug along.
I193	Culture; Process; Release	It's the working together of people doing development tasks and operations tasks to keep the common goal of software, as it's being produced, being brought out into the production environments in a kind of way of working together.
I202	Culture; Resistance	It was just frustration down the line that we didn't really find a way of working together on anything other than first line support which I think, to be fair, we've now found a way. I think the problem then is it leads to, 'well we've tried this and we've tried this and we've tried this and it's all sort of, no we can't do it' so therefore we don't really try to engage particularly with things we've been told. We don't want to engage with that so I think we've found this nice balance at the moment with first line support, they're both quite happy

		with that. I think <name omitted> said explicitly that's where it starts and ends. So we both know where we are. So the frustration is now gone.
I206	Culture; Resistance	The main challenge is to get the key people in the business to sort of buy into this workflow at the right time and not to say, 'I haven't got time', 'I'm too busy' or need to sort of be there. So it's to get people outside the immediate DevOps type environment to buy into it working. I think they certainly like it when we do it. It's certainly not how it works currently.
I210	Culture; Business Management	If we have Azure the Developers were dealing with it, but as they're going back to in-house now, it's going to go back to Support, so technically they're in the same position as when they started.
I212	Knowledge Management; Transformation	Personally I think the skill set of the whole team has gone up and that's mainly through the feedback via call requests and this gives the opportunity to share some skills and knowledge and techniques between say new employees and existing ones who haven't been on the system before. So I think that's very, very positive.
I214	Legacy Systems; Quality	Code quality matters in any environment. If it becomes unmanageable or too complicated instead of taking five minutes to fix, it takes five days. So that's the situation with AFI, to actually do anything took you longer to undo the bugs that the change caused!
I222	Job Crafting	We feel like we've got a bit more freedom to go and – 'We want to switch on this feature and have a look and see what it does'. Even though we're going to end up hosting in-house, we're still using the cloud for development and testing, which is great because that frees up a lot of bottlenecks in our development process.
I233	Legacy Systems; Business Management	There's definitely a set of AFI work coming but it'll be a temporary thing and then at some point it'll be switched off or it'll just be left alone probably for historical reasons.
I234	Process	We've got a support team that should be supporting the floor and then we have our development team that is actually building, delivering and releasing the software out to the system, to the clients.
I236	Decision Making; Business Management	They are making the decisions on what we're doing, how they want the system to be built and how they want the system to work and just specifying any additional rules that we don't know. It would be nice to have more input on the bigger areas of making the decisions, but yeah, senior management doesn't really worry too much about it.
I237	Legacy Systems; Quality; Process	Well, I was shocked actually they were doing so many workarounds outside the system in order to get the information into the system, and that's what was shocking. But then, when you're trying to build Harrier to encompass all the rules, you can kind of feel why they're doing everything out of the system, because there's so many business rules based around that we're now having to program and actually put into Harrier, whereas in AFI it wasn't there at all.
I238	Process; Decision Making	The person who holds all the information needs to be involved basically, but it does worry us because we're not able to move certain areas forward until we've got various answers and time, because the person who has all those answers has very tight time. It will delay the project completely and if we want to keep moving forward and try to hit some kind of deadline, then we need more involvement.
I245	Culture	They usually go through me if we need anything between Dev and Support.
I247	Job Crafting	They're starting to engage, but I think it's from the demo. So as soon as they've seen what we actually have done, what we've actually produced, they've become more involved and want to see the end of it.

Appendix 13: Specimen DevOps Engineer Job Description

Example job description for a DevOps Engineer, taken from an advert posted by Adams [2019]

Key Accountabilities/Responsibilities:

- Designing and developing scripts/tools for Continuous Integration and Deployments.
- Designing and developing automation templates/tools for infrastructure provisioning, configuration & change management.
- Building and deploying web applications to dev/test/prod environments.
- Automating configuration management, infrastructure and application deployments in a toolset such as Puppet.
- Own, manage and improve our release process. Focus on scale and efficiency.
- Work with Operational and Development groups to drive the most optimal solutions.
- Work in a fast-paced dynamic environment.
- Create and maintain documentation for the solutions provided.
- Communicate with stakeholders and peers from different areas of our Businesses, tailoring messages to the targeted audience.
- Work with Engineers and Analysts on software builds and deployment troubleshooting.
- Demonstrate thorough understanding of major system components (i.e., storage systems, Linux kernel, UNIX kernel, UNIX file system, and Windows infrastructure).
- Configure controls; install and troubleshoot applications.
- Work closely with relevant Technology groups to refine system monitoring and reporting.
- Collaborate with the test team to ensure test validation and release management principles are upheld.
- Apply problem-solving skills to support assignments.
- Diagnose system performance problems.Promote and support agile working and DevOps methodology.
- Develop scripts for execution of commonly used processes and automation of simple tasks.

- Creation, execution, documentation and completion of tasks, changes, and requests.
- Collaboration and teamwork; actively develop strong, supportive and collaborative working relationships.
- Apply technical expertise to support strategic decisions and thought leadership.
- Support Development teams using development tools, products and processes.
- Continue expanding and improving our DevOps delivery pipeline.

Experience/Skills Required:

- Well versed in Puppet or Ansible
- Experienced in using GitLab
- Experienced in infrastructure as code tools, such as Terraform (including XML and JSON type configurations)
- Terraform
- Experienced in automated build and deployments using Jenkins / GitLab CI
- Programming / Scripting (PowerShell, Bash, or Python)
- Jira; Configuration, Administration and Scripting skills
- Working knowledge of containers (Docker, Kubernetes etc)
- Operating Systems including Windows, Linux/UNIX
- Enterprise level networking (TCP/IP, VPNs, SFTP, Proxy, Firewalls)
- Strong communication and collaboration skills
- Excellent problem solving skills
- Experience with end to end Continuous Integration and Continuous Deployment pipelines

Glossary

Abductive Reasoning A form of logical inference starting with an observation before seeking to find the simplest and probable explanation for it.

Agile An umbrella term for a set of methods and principles where solutions evolve as a result of the collaboration between customers and developers in self-organising and cross-functional teams.

API An 'Application Programming Interface' is a set of functions and procedures that enable a software applications to access the features or data of another.

Axiology The study of the nature of value and valuation, including the kind of things that are valuable.

Back-End Developer A software developer who implements core and computational logic components of a software system that are indirectly accessible to users through a front-end.

Case Study A research strategy involving the empirical investigation of phenomena within its natural or real-life context.

Constructivism Ontological position asserting that reality is subjective and a mental construct by individuals through cognitive and social interraction processes.

Continuous Deployment (CD) A process of minimising lead time in the delivery of software. For example, the time between a line of code being written to that same line of code behind deployed as part of live software.

Continuous Integration (CI) The practice of merging source code into a shared branch frequently, often several times a day, for the purposes of automated testing and identification of issues.

Coupling Describes the interdependance of software components. It is often referred to how easy software maintenance can be, as in the case of loose couping. Conversely, highly coupled software is often considered far more difficult and risky to maintain.

Deductive Approach An approach that tests theoretical propositions through employing a specifically designed research strategy and subsequent methods for such testing.

DevOps A portmanteau of 'development' and 'operations', specifically referring to an organisation's software development and IT operations functions.

Eptistemology The branch of philosophy that studies the nature of knowledge and what constitutes acceptable knowledge in a field of study.

Fallibilism A principle that postulates empirical knowledge can be acceptable even if it is unable to be proven with certainty.

Front-End Developer A software developer who implements components of a software system that are directly accessible to users.

Full Stack Developer A software developer who works across both the front-end and back-end when implementing features within a software system.

Git (Software Engineering) A widely used protocol for the version management of software source code, allowing the coordination of work on multiple files across multiple people and teams.

Git Add A git command used include a recent change to the 'staging area', indicating to Git that you intend to include said change in the next commit. This is also known as 'staging' and is always the precursor for using the Git Commit command.

Git Commit A git command used to 'commit' added or 'staged' changes to the project on the developer's computer. A single commit can include one or more staged changes.

Git Pull A git command used to retrieve and merge the latest source from the remote repository (server) with the version a developer is working with locally on their computer.

Git Push A git command used to send and merge committed changes from a developer's computer to the remote repository. One or more commits can be included in a single Git Push command.

Inductive Approach An approach that develops theory as a result of research activity.

IT Operations A branch of Operations Management concerned with the continuity of business IT infrastructure and provision of support (helpdesk service) for both hardware and software issues.

Job Crafting A theory proposed by Amy Wrzesniewski and Jane Dutton [2001] that describes the ways in which employees customise their jobs by the active changing of tasks, relational and cognitive boundaries of their work.

Kanban An increasingly popular cyclical Agile framework and workflow solution for dealing with especially complex or chaotic user requirements in software engineering projects. Strongly associated with Toyota and with roots in lean manufacturing, Kanban places emphasis on demand and available capacity in addition to the simple and clear visualisation of workflows.

Markdown Lightweight markup language specifically taking an easy to read plain text format, converting it to HTML when rendered by a web browser.

Microservice A software development technique and architecture which arranges a software application as a collection of loosely coupled services.

Mixed-Method Research The employing of both qualitative and quantitative methods of data collection and analysis either at the same time or in sequence.

Multi-Method Research The employing of either more than one qualitative or more than one quantitative method of data collection and analysis.

Ontology Branch of philosophy that studies the nature of reality or being.

Positivism Ontological position asserting that reality is objective and external to an individual. The epistomology asserts that social realities can be externally measured through a deductive approach involving highly structured methods, including hypothesis testing, leading to law-like generalisation.

Pragmatism Ontological position arguing that research questions are the most important determinant of the research philosophy and subsequent approach and strategy taken.

Provisional Truth The belief that knowledge, meaning and truth is tentative and subject to change both over time and at any time.

Research Approach Generic term referring to a deductive or inductive approach.

Research Philosophy Overarching term concerning the nature of reality (ontology) as well as the development and nature of knowledge (epistemology) in relation to research. The research philosophy taken often dictates the approach, strategy, data collection methofd and time horizons of a research project.

Research Strategy The general plan underpinning the manner in which a researcher will go about answering their research questions.

RESTful Representational State Transfer (often shortened to REST) is an architectural principle with web applications allowing the requesting and receiving of data. First defined within the PhD work of Roy Fielding [2000].

Retrospective Bias The position of seeing an event after it has happened as having been predictable; also known as hindsight bias.

Scrum A well defined Agile software development framework credited to Ken Schwaber and Jeff Sutherland [Rubin, 2012]. Scrum is widely used in the development of software, where developer activities are timeboxed into small sprints (usually over a matter of weeks) with the goal of producing either a working software artifact or viable increment.

Software Crisis A period in the mid 20th century coined to describe the lack of any formal approaches to software development despite additional problems introduced from increasing complexity, maintenance and technological innovation.

Triangulation Involving and using multiple sources so as to enhance the rigour of research activities.

Waterfall First structured software development approached credited to Royce [1970]. Waterfall denotes a linear and rigid series of steps which must be completed in order. It was widely adopted but also heavily criticised for its rigidity and presumption of user behaviour.