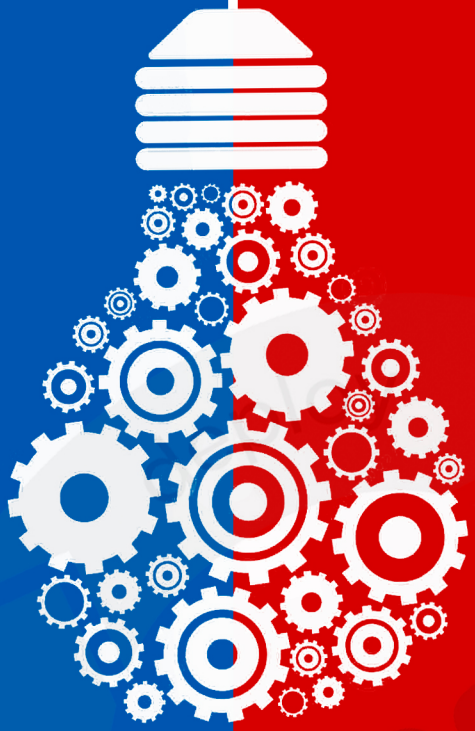
Nikolaj Falk Jonassen (101613)
Niklas Sachs Tautum (101824)

CBS

The Never-Ending Process of Improving DevOps Maturity

A longitudinal paired-case study of maturing DevOps within organizations



Master's Thesis
Cand.Merc.IT
107 pages / 229.897 characters

Supervisor: Till Winkler

Copenhagen Business School
May 15 2020

Abstract

The concept of DevOps, a cultural movement and technical solution that combines development and operations, has gained considerable interest from practitioners since its introduction 2009. The increased attention is grounded in the exceptional value proposition that DevOps promises. However, organizations still lack practical evidence on the adoption and maturing of DevOps. Motivated by the increased attention on DevOps, this thesis investigates how organizations can mature their DevOps approach through a longitudinal paired-case study of two Danish organizations. Based on 26 interviews with various IT professionals, we adopt a maturity model and process theory to facilitate a broader understanding of DevOps and the processes within. We not only identify several drivers and capabilities that can mature an organization's DevOps approach but also contribute to the existing research by defining crucial challenges and pitfalls associated with DevOps. Our research pioneers the discussion of whether organizations can become too DevOps mature. This discussion contributes to a critical view of maturity models that may assist individual organizations in assessing the optimal level of maturity.

Preface

Reference Standard

When referencing literature, this thesis will utilize the American Psychological Association (APA) standard. The APA standard consists of the following structure: (Author, Year of publishing). This standard is very widely used within research, and by the vast majority of the literature included in this thesis, why this thesis will utilize the same method for consistency and readability. A comprised list of the literature that has been referenced in this thesis will be available in the bibliography at the end of the thesis.

When this thesis references interviews, both from the primary and secondary data, the reference will appear in the following way: (Respondent X, Organization, Year). A list of all respondents, with corresponding numbers, is found in section *11 Appendices*.

Acknowledgments

We want to extend our sincerest gratitude towards the supervisor of the thesis, Till Winkler, for his availability and guidance throughout the process of writing. Furthermore, we express our gratefulness to the two case organizations, Topdanmark and Proactive, for allowing us the opportunity to perform a case study within the companies and delegating resources to the cause. The close collaboration between the researchers and each of the case companies is much appreciated. Lastly, we express our gratitude towards Associate Professor at the IT University of Copenhagen, Oliver Krancher, for his valuable inputs and guidance in shaping the thesis.

Table of Contents

Preface 1

Reference Standard 1

Acknowledgements 1

1 Introduction 6

1.2. Research Question 8

1.4. Structure of Thesis 9

2 Literature Review 12

2.1. Literature Review Method 13

2.2. Fundamentals of DevOps 15

2.3. DevOps Challenges 17

2.4. DevOps Culture 18

2.5. DevOps Value 21

2.6. Software Maturity Models 24

2.7. DevOps Maturity Models 26

3 Methodology 31

3.1. Research Philosophy 31

3.2. Research Approach 32

3.3. Research Design 33

3.4. Longitudinal Analysis 35

3.5. Data Collection 36

3.5.1. Operationalization 36

3.5.2. Primary Data 37

3.5.3. Secondary Data 40

3.5.4. Referencing Interview Quotes 41

3.6. Data Analysis 41

4 Case Descriptions 43
4.1. ProActive 43
4.2. Topdanmark 44
5 Analytical Framework 46
5.1. Maturity Assessment 46
5.2. Process Theory 47
6 Within-Case Analysis 49
6.1. ProActive Maturity Assessment 49
6.1.1. Previous Level of Maturity 49
6.1.2. Current Level of Maturity 51
6.1.3. Summary 58
6.2. Topdanmark Maturity Analysis 59
6.2.1. Previous Level of Maturity 59
6.2.2. Current Level of Maturity 62
6.2.3. Summary 70
6.3. Visual Maps 72
6.3.1. Issue Domains 72
6.3.2. Boxes 73
6.3.3. Direct and Indirect Relationships 73
6.3.4. Maturity Indicators 74
6.4. Visual Map of ProActive 75
6.5. Visual Map of Topdanmark 76
7 Comparative Analysis 77
7.1. Procedural Analysis 77
7.2. Challenges 80
7.3. Findings 85

8 Discussion 88
8.1. Organizational Size 88
8.2. Organizational Structure 88
8.3. Tools and Infrastructure 91
8.4. Communication 93
8.5. Tenure, Competences, and Mindset 94
8.6. Fragmented Planning Activities 95
8.7. The Perceived Value of DevOps Maturity 96
8.8. Implications for Theory 100
8.9. Implications for Practice 100
8.10. Limitations 101
8.11. Future Research 103
9 Conclusion 105
10 Bibliography 107
11 Appendices 112
11.1. Appendix A – Competence Model (Feijter et al. 2018) 113
11.2. Appendix B – Focus Areas and Capabilities (Feijter et al., 2018) 114
11.3. Appendix C – Interview Guide 120
11.4. Appendix D – Primary Data 122
11.5. Appendix E – Secondary Data 123

Table of Figures

Figure 1: Structure of thesis 11
Figure 2: The fundamentals of DevOps 16
Figure 3: The Topham Model by Inbar et al. (2013) 26
Figure 4: The maturity model by Mohamed (2015) 27
Figure 5: The Focus Area Model by Feijter et al. (2018) 29
Figure 6: Snippet of the interview guide 39
Figure 7: Assessment of ProActive's previous level of maturity (2015) 50
Figure 8: Assessment of ProActive's current level of maturity (2020) 52
Figure 9: ProActive's DevOps maturity progression 58
Figure 10: Assessment of Topdanmark's previous level of maturity (2015) 60
Figure 11: Assessment of Topdanmark's current level of maturity (2020) 63
Figure 12: Topdanmark’s DevOps maturity progression 70
Figure 13: Visual map of ProActive 75
Figure 14: Visual map of Topdanmark 76
Figure 15: The fully embedded topology (Skelton & Pais, 2019) 90
Figure 16: The DevOps team silo (Skelton & Pais, 2019) 90

Table of Tables

Table 1: Concept matrix 12
Table 2: Overview of operationalization 37
Table 3: Overview of respondents (primary data) 38
Table 4: Overview of respondents (secondary data) (Nielsen et al., 2017) 40

1 Introduction

Previously, the traditional approach to software development included a distinct handover of code from development to operations, enforcing clear work and cultural boundaries between the two. Now, the growing need for the ability to release new applications, features, and bug fixes daily has led many organizations to explore new strategies for software development. Building on lean and agile practices, the DevOps concept has emerged and significantly impacted the entire IT and software industry with the promise of end-to-end automation of software development and delivery (Ebert et al., 2016). DevOps is derived from the combination of the two words, development and operations, and intends to blend these practices into one efficiently operating cohesion to overcome the traditional boundaries (Lwakatare, Kuvaja, & Oivo, 2015).

Research has revealed that IT organizations experience staggering amounts of cost due to unplanned downtime in applications (Elliot, 2014). Furthermore, Elliot (2014) reports that, on average, 25% of an application's development and operations life cycle is considered wasteful and unnecessary. These are some of the problems scholars believe that DevOps can diminish. Ebert et al. (2016) report that the successful adoption of DevOps improves cycle times by 10 to 30% while reducing cost by up to 20% due to mutual understanding of requirements, maintenance, service, and product evolution. Though DevOps creates benefits, it also brings particular challenges for companies. The challenges of achieving DevOps include the integration of a supportive technical architecture and a shift in culture and mindset (Ebert et al., 2016).

Without the combination of a supportive technical architecture and an embedded DevOps culture, organizations seeking to adopt DevOps cannot function effectively (Elliot, 2014). Walls (2013) emphasizes this exact combination as the key to successfully adopting DevOps. Walls (2013) highlights DevOps as a cultural movement combined with software development practices, in which organizations tend to neglect the cultural element. Feijter et al. (2017) report similar findings and stress the need for the effective combining of technical and cultural aspects.

Streams of literature have emerged in an attempt to address, explore, and explain potential approaches for the successful adoption of DevOps. Some draw on the traditional theory of maturity models, seeking to identify the capabilities and characteristics that mature software organizations utilizing

DevOps have (Mohamed, 2015; Feijter et al., 2018; Inbar et al., 2013). Collectively, these studies provided an understanding of the capabilities and characteristics necessary for organizations to achieve maturity in DevOps.

However, DevOps as a research concept is somewhat new; as such, the existing amount of literature on DevOps maturity and adoption is somewhat scarce and limited in terms of practical evidence. The currently available literature on DevOps maturity has several limitations, such as models lacking empirical validation in terms of accuracy and applicability due to the low number of cases investigated (Feijter et al., 2018). The lack of specificity is further a limitation as the existing maturity research puts limited emphasis on which areas and events in the DevOps maturing process that are most impactful on organizations (Zarour et al., 2019). Existing research has shown DevOps to be a complicated measure, causing these types of cookbook-style models to be challenging to follow, as they do not account for the individual characteristics of the investigated organizations (Ebert, 2016). Moreover, as the general assumption of maturity models is that it is always favorable to progress maturity, the existing research does not account for the aspect of organizations becoming too DevOps mature (Zarour et al., 2019; Gasparaité & Ragaišis, 2019).

This thesis will study the maturity process of organizations by analyzing the adoption of DevOps by two Danish organizations over five years. The analysis covers an assessment of the previous and current levels of maturity, in addition to challenges and benefits related to the adoption of DevOps. In the analysis, we apply the latest research to understand the underlying concepts and capabilities associated with the concept of DevOps.

For the assessment of the investigated companies' maturity level, we adopt the maturity model developed by Feijter et al. (2018) as our analytical framework. Additionally, we utilize process theory for establishing an overview of how the two case organizations have approached DevOps. As part of our research, we investigate commonalities and differences across the two cases to find results that can contribute to the limited knowledge base of DevOps. Lastly, we discuss the possibilities of whether organizations can become too DevOps mature.

1.2. Research Question

Addressing the presented limitations of current research, the question framing this thesis is:

How can organizations mature their DevOps approach?

To answer the research question adequately, we address the following sub-questions:

- What are the drivers and capabilities that progress DevOps maturity?
- What hinders organizations from maturing DevOps?
- Is it possible for organizations to become too DevOps mature?

1.4. Structure of Thesis

The following section presents the structure of the thesis. We describe each section and finally visualize the structure in a model to form an overview of the thesis.

Section 1 introduces the thesis and the limitations of current research that we have identified in the investigation of the topic. This section also presents the specific research question and the related sub-questions that together serve as the focal point of the research.

Section 2 presents the existing streams of literature on the investigated topic of DevOps. Here, we explore previous literature on the fundamental characteristics of DevOps, associated challenges, and the value-adding activities related to DevOps. Furthermore, the aspect of maturity is introduced by explaining the origins of software maturity models. In connection with this, we review current maturity models associated with DevOps. This section also documents the methodology used to establish the literature review, including the methods used to find relevant research and create a concept matrix.

Section 3 explains the methodology and the paradigmatic basis of the research. Initially, the section describes the paradigmatic and epistemological approach to the study. Subsequently, we present an introduction to the research design and the longitudinal perspective involved in the thesis. As the other decisions related to the empirical approach are rationalized based on these choices, it is natural for this to be determined as the first part of the methodology. Furthermore, we describe the operationalization process that serves as a link between the theoretical and empirical levels. Finally, we present the empirical considerations, including the empirical sources, the data collection techniques, and the analysis of the collected data.

Section 4 offers a presentation of the two investigated case organizations. This entails a brief description of each organization's history, the industry where they are situated as well as their approach to software development.

Section 5 presents the analytical framework utilized in the analysis. This includes the adopted maturity model for assessing the maturity levels of the two case organizations and an introduction to process theory and the used sensemaking strategy (visual mapping strategy).

Section 6 consists of the first part of the analysis, a within-case analysis. The within-case analysis is composed of an assessment of the previous and current levels of maturity within the two organizations. This is followed by two visual maps, one for each organization, to highlight the temporal aspect of the DevOps maturing process.

Section 7 consists of the second part of the analysis, a comparative analysis. The comparative analysis includes a procedural analysis of commonalities and differences found in the within-case analysis. Furthermore, we compare the challenges experienced by each organization. Lastly, the findings from the analysis are summarized.

Section 8 consists of a discussion of the findings. The findings are compared with the previous literature and critically evaluated. Besides, we discuss whether an organization can become too DevOps mature. This section also contains the implications for theory and practice, and an introduction to the possible limitations of the research. Finally, we present possible areas for future research.

Section 9 concludes the thesis with an answer to the research question, we initially presented.

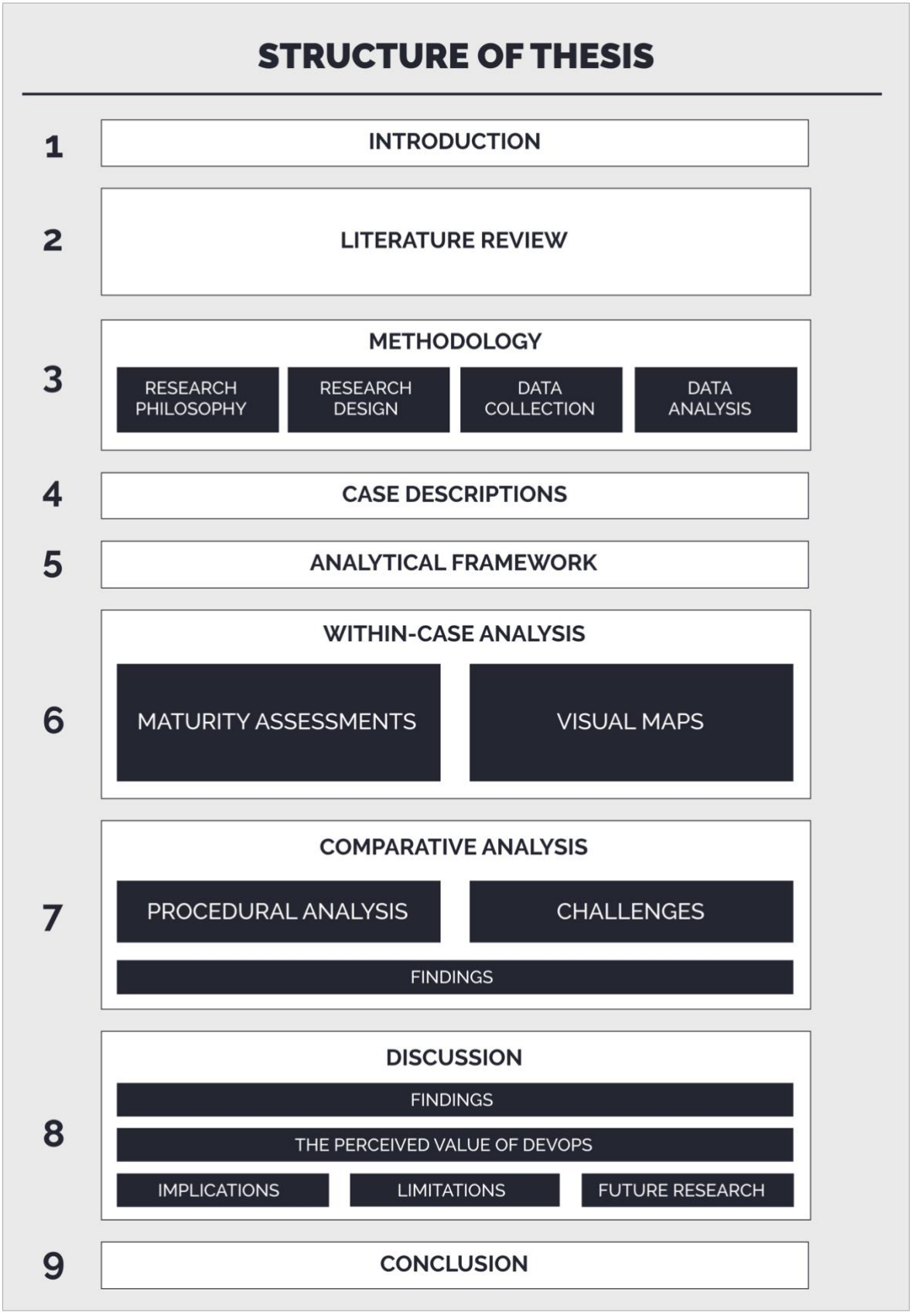


Figure 1: Structure of thesis

2 Literature Review

In this section, we review previous research on DevOps and maturity models, and the underlying concepts of both, concerning their implications for organizations within the field of software development. This literature review will utilize a concept-centric approach that will allow for a systematic method to review several relevant themes. The themes are closely tied to the concept of DevOps and have shown continuously throughout the literature. Within recent years, DevOps has steadily become more articulated in literature, yet there are still complex and incongruous elements tied to DevOps that emphasize the necessity for a concept-centric approach. The concept-centric approach provides the reader with a comprehensive and fulfilling depiction of DevOps and the underlying concepts. The use of a concept-centric approach will ensure that all relevant aspects of the DevOps concept are divulged, which in turn will establish a firm foundation for advancing knowledge (Pfeffer & Sutton, 2006).

Table 1: Concept matrix

Author(s)	Concepts				
	DevOps Fundamentals	Challenges	Culture	Value	DevOps Maturity
Cusick (2019)					X
Gasparaite & Regaisis (2019)					X
König & Steffens (2018)	X		X	(X)	
Becker et al. (2009)					(X)
Ebert (2016)	X		X	X	
Feijter et al (2017)	X	X	X	X	X
Feijter et al (2018)	X	X	X	X	X
Mohamed (2015)	X	X	X	X	X
Mohamed (2016)	X			X	X
Lwakatare et al. (2015)	X		X		
Walls (2013)	X	X	X	X	
Aiello & Sachs (2016)	X			X	X
Reed (2014)				X	
Edwards (2010)	X		X		

Krancher et al. (2018)				X	
Elliot (2014)	X	X		X	
Gill et al. (2017)	X	X	X	X	
Dingsøyr & Lassenius (2016)			X	X	
Wiedemann et al. (2019)		X	X	X	
Kim (2018)		X	X	X	
Ghantous & Gill (2017)		X	X		
Riungu-Kalliosaari et al. (2016)		X	X		
König & Steffens (2018)			X		X
Lu & Ramamurthy (2011)		X		X	
Virmani (2015)	X		X	X	
Zarour et al. (2019)		X			X
Inbar et al. (2013)		X	X	X	X

The chosen literature used to encompass the concept of DevOps, and the use of maturity models within, is presented in the concept matrix in *Table 1*. The matrix includes the chosen concepts that have shown continuously throughout the literature. The concepts are carefully chosen due to their importance, relevance, and impact on the adoption and maturing of DevOps.

2.1. Literature Review Method

We review a substantial amount of current literature of DevOps to conduct a comprehensive and fulfilling literature review. To locate the research chosen for further investigation and included in this literature review, we have utilized selected databases. The primary literature databases used in finding relevant research have, in this thesis, been Google Scholar and CBS Libsearch. We utilize other databases such as Research Gate, Elsevier, and Jstor as secondary databases. These have been utilized to find specific articles or used for their "related articles" feature that has shown to uncover articles with relevance in the discussion of DevOps.

According to Jalali & Wohlin (2012), if a systematic literature review begins with an offset in database searches, some relevant keywords, connected to the investigated topic, should be defined.

These keywords should then be included when searching through several databases. Typically, a useful and relevant publication is published in one of the more prominent journals or well-known conferences as these have a reputation of quality. According to Jalali & Wohlin (2012), the initial phase of database searching will reveal a substantial amount of literature within the field of the investigated topic. The available research will then need to be filtered to fit the specific research topic. Due to the possible sizeable amount of research, articles should be hastily reviewed on included keywords or by abstract to determine relevance for the specific research topic. It is proposed by Webster & Watson (2002) that the review of the literature connected to the researched topic is close to finalization when the researchers no longer are being introduced to new concepts in the articles.

In the search for relevant literature for use in this thesis, we include the following keywords in the searches on Google Scholar and CBS Libsearch: DevOps, DevOps culture, DevOps value, "Maturity models", "DevOps Maturity models", DevOps challenges, and "Continuous Delivery". Note that quotation marks encompass some of the keywords to create key phrases. Had the keywords not been encompassed by quotation marks, then the databases would have searched for each word in the phrases by itself, and would therefore not have the same meaning or relevance since the databases would locate anything including either of the words.

The database search approach was utilized in the initial phase to locate literature for the theoretical grounding of this thesis. However, due to DevOps being a relatively new and comprehensive concept, the amount of relevant literature was not overwhelming. This is not to be misunderstood with there not being a substantial number of articles that include the somewhat popularized word of DevOps, but the number of articles that comprehensively researched and wrote about DevOps in detail was relatively scarce.

In contrast to the database search, Webster & Watson (2002) propose a slightly different approach to systematic literature reviews in the field of information systems. They propose to use a method reminiscent of snowballing as the primary method to find literature. Snowballing is a known technique from systematic literature reviews, where the search for literature is based on the reference list or citations of a research paper to identify additional research (Wohlin, 2014). However, instead of using the snowballing technique in the first phase of the literature search, we have applied it in the later phases, as we found relevant literature through the databases. Webster & Watson (2002)

advocate using both backward and forward snowballing. Backward snowballing consists of finding additional literature from the reference list of the already chosen articles. The forward snowballing approach is concerned with finding citations to the papers. This thesis has utilized both backward and forward snowballing in the later phases of locating literature, but backward snowballing has been most prominent.

2.2. Fundamentals of DevOps

Software companies that offer internet-based services or Software as a Service (SaaS) have now mostly transitioned away from the traditional delivery methods that were characterized by extensive functionality deliveries, delivered with a specific interval (e.g., monthly, quarterly or bi-annually). Now, a lot of these companies can and strive to deliver a solid stream of smaller functionality daily instead of delivering substantial updates of functionality in the predetermined interval (Ebert et al., 2016).

This paradigm change towards continuous delivery and continuous deployment of software functionality brings both challenges and opportunities for most companies (Lwakatare et al., 2015). The concept of DevOps was introduced by Patrick Debois, known as "The Father of DevOps", to facilitate this paradigm change. DevOps was initially presented at the DevOps Days conference in September 2009 (Mamatha & Kiran, 2018), but the concept was not entirely new to the field. DevOps was born primarily due to the increasing adoption of cloud services that changed the way software development traditionally worked (Smith, 2011). DevOps is a contraction of the two words, Development and Operations, and it seeks to break down the traditional silos between the development teams and the operations teams in companies by integrating the two worlds using automated development, test, deployment, and monitoring.

While DevOps resides within IT organizations and is directly tied to technology, it is, first of all, an organizational shift in culture, where instead of having development, quality assurance (QA), and operations teams performing functions separately, cross-functional teams are created with a focus on continuous feature deliveries (Ebert et al., 2016).

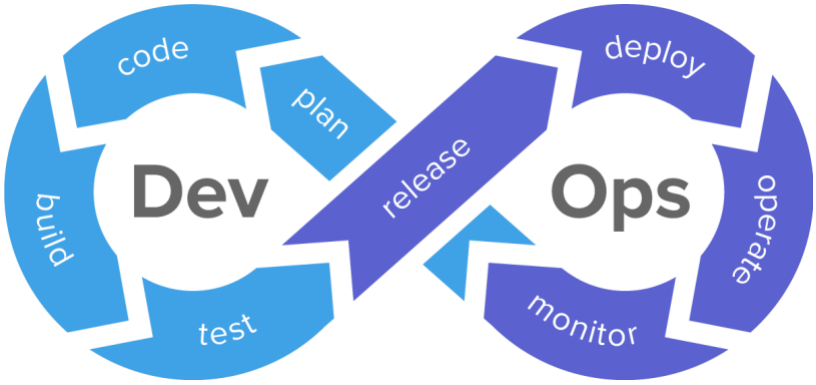


Figure 2: The fundamentals of DevOps

DevOps establishes a culture of end-to-end responsibility of functions and features that ensures system compatibility of developed code and, thereby, mitigate the risk of encountering traditional problems. The traditional problems are often tied to code "being thrown over the wall" and not seen again by developers until the code does not build or function in the system on the date of delivery, where everything is compiled. If carried out successfully, DevOps help to deliver value continuously in a faster and more consistent manner while reducing problems tied to miscommunication between team members as well as hastening problem resolution (Ebert et al., 2016).

To reach a level of success, however, DevOps requires companies to increase communication amongst stakeholders, implement automation, and improve agility in designing, delivering and operating software products and services (Lwakatare et al., 2015). Furthermore, a high degree of automation is needed to make quality deliveries with short cycle times, which extends to a mandatory need for tools in DevOps. As such, choosing and utilizing the right tools is crucial for any organization that seeks to adopt a thriving DevOps culture.

DevOps can be applied to extremely varying delivery models but needs to be tailored to the individual environment and product architecture (Ebert et al., 2016). Thus, the tools that underlie the automation services of DevOps, chosen by the individual organization, need to reflect the environment and product architecture of that organization. These tools need to be integrated into the complete process from build and continuous integration to logging and monitoring.

2.3. DevOps Challenges

Within the existing research of DevOps, many different challenges have emerged and shown to have a direct impact on the success of a DevOps implementation (Elliot, 2014; Ghantous & Gill, 2017; Riungu-Kalliosaari et al., 2016). Of these challenges, the culture and mindset within the organization are regarded as predominant to the success of an organization's approach to DevOps. Shifting the culture to support the DevOps ideal is paramount, and cultural inhibitors that block collaboration within teams will hinder the organization from achieving the actual value of DevOps. Merging roles, sharing responsibilities, and rethinking the daily workflows are among some of the initiatives that make DevOps seem scary to the majority of team members (Ghantous & Gill, 2017). This thesis adopts the view of DevOps culture as a vital factor in the implementation and use of DevOps, for which reason the cultural aspect of DevOps will be further elaborated separately in the section *2.4 DevOps Culture*.

Another commonly recognized key impediment that organizations must be aware of is the communication within the team. Organizations often experience insufficient communication between the development and operations teams, which will produce detrimental outcomes. The reduced flow of information can be traced back to the predefined roles in which the team members still are ingrained. For instance, developers still care more about release frequencies, whereas operations still focus on the stability of the system, leading to a disjointed process (Riungu-Kalliosaari et al., 2016). Moreover, communication is often formal and done through documents or emails due to the separation of team members. Removing formal communication channels will ease the communication as the information flow can move a lot faster, while simultaneously fostering and establishing a stronger relationship between the development and operations (Walls, 2013).

Another obstacle presented by previous research is the fragmented planning activities that occur during the development process. Development and operations often do not cooperate when new development projects are about to be launched, which means that the requirements, as well as the expectations for a given feature, might not align between the two teams. For instance, operations are, most of the time, not included in the initial phase of the development, resulting in negligence of system stability or other system requirements (Elliot, 2014). Moreover, fragmented planning activities challenge the automation of the software development process. Due to operations not being

initially included, it forces a distinct handover, which might limit the operations team's ability to implement automated practices such as testing, monitoring, and deployment (Ghantous & Gill, 2017).

Riungu-Kalliosaari et al. (2016) highlight heterogeneous environments and immature infrastructures as obstacles that can challenge organizations in adopting DevOps successfully. Heterogeneous environments are characterized by a high degree of complexity. They are often comprised of components from different vendors, making it very difficult to build replicates of these types of environments for test and release scenarios. Consequently, automated practices become unreliable as tailored implementation measures are necessary, forcing teams to consistently maintain components that otherwise would need less attention (Riungu-Kalliosaari et al., 2016). In continuation of this, Ghantous & Gill's (2017) research identified tools as another significant challenge as development and operation teams use completely different toolsets and metrics. While the clash between tools most likely roots back to the resistance to change, teams should aim to reduce the number of must-haves and align the infrastructure to support the overall goals of the organization.

Prior research also identifies the size of the organization as a barrier that can hinder the adoption of DevOps. Generally, smaller organizations have an advantage over larger ones when it comes to change management, as they can fine-tune resources in order to influence the behavior of the individual team members (Walls, 2013; Riungu-Kalliosaari et al., 2016). Besides, small organizations have a more favorable position to react faster to changes in the environment, providing them with a competitive advantage (Riungu-Kalliosaari et al., 2016). However, having defined teams that can support the change might help larger organizations mitigating the risk of incurring challenges due to size.

2.4. DevOps Culture

As discussed, DevOps is not solely a question of technology, but also a question of culture. According to Walls (2013), DevOps is as much about culture as it is about tools. In more detail, Walls (2013) refers to DevOps as "being a cultural movement combined with a number of software development practices that enable rapid development". A viewpoint that contains many similarities with the one of Feijter et al. (2017), presenting six drivers towards DevOps, the first being "a culture of collaboration". The importance of encompassing a culture of collaboration is stressed in the vast

majority of available academic literature as one of the most fundamental things when adopting DevOps (Ebert, 2016; Feijter et al., 2018; Lwakatare et al., 2015).

The reason behind the importance of establishing a thriving DevOps culture is due to the breaking down of silos and the transition from the traditional teams to the new, cross-functional teams (Lwakatare et al., 2015; Feijter et al., 2017). Bridging the gap between development and operations, and other stakeholders such as testing and QA teams are often not perceived to be the most challenging task when adopting the DevOps concept. However, according to Ebert (2016), it is something that most organizations underestimate, leading to the entire initiative going awry. When an organization attempts to bridge the gap between the Dev and Ops teams, and perhaps other teams, it is not sufficient to simply suggest a culture of collaboration. Instead, a genuine effort must be made to actively change the mindset of the affected people (Lwakatare et al., 2015; Walls, 2013). The individual teams have, like most other independent teams, gradually created their own shared culture of performing work and communicating.

The procedures of which a development team and an operations team would undergo in processing and solving an identical problem is most likely very different, even though the two teams work in the same organization. The challenges in creating a culture of collaboration between two distinctive teams are partly due to the mindset, but also the team's perception of the other team has an influence. The operations team might perceive the developing team as cumbersome and exasperating, and contrariwise the development team perceives the operations team as being quick, dirty, and unscrupulous (Ebert, 2016). As such, adopting a DevOps culture is complex and essential for the success of the DevOps initiative. The available research and literature are in concurrence concerning the importance of culture within DevOps, but in some quarrel regarding what elements and aspects are involved in a thriving culture.

According to Walls (2013), talking about culture in absolute terms and endeavoring to generalize and create a formula of a thriving DevOps culture is ludicrous, because of no two groups, and thereby cultures, being alike. The view of Walls is opposed by Feijter et al. (2017), however, accentuating that while there is no universal and standardized solution for the perfect DevOps culture, a general guideline can be created. The general guideline would include a set of aspects and capabilities that

are important for a DevOps culture, and would then be tweaked and tailored to the individual organization (Feijter et al., 2017; Feijter et al., 2018).

Within the developed competence model of Feijter et al. (2017; 2018), culture and collaboration are boiled down to five focus areas: Team Organization, Communication, Trust and Respect, Knowledge, and Release Alignment (please refer to *Appendix A*). According to the researchers, the area of culture and collaboration is the most prominent of the model. The focus areas of Feijter et al. (2017; 2018) are rather self-explanatory but revolve around creating a culture that enables teamwork, knowledge sharing, and alignment between internal and external dependencies to timely deploy software.

While Walls (2013) does not advocate for a generalized guideline of culture, she has proposed four vital cultural characteristics that can help to mitigate friction between the development and operation teams when they are united. The characteristics of Walls (2013) are, to some extent, similar to those of Feijter et al. (2017; 2018). The four characteristics are Open Communication: Incentive and Responsibility Alignment, Respect, and Trust. The first characteristic of open communication is articulated by Walls (2013) as being fundamental for a DevOps team.

A DevOps team needs to be able to discuss the product regarding requirements, features, schedule, resources, production, build, and many other areas. The second characteristic of incentive and responsibility alignment is centered around motivation and reward for team members of the DevOps team. An optimal team unites around the core goal of the organization, which should be to create the best product for customers of the business. As such, developers should not be rewarded for delivering many lines of code and impressive features, and operations should not be punished when the code does not run as expected in a production environment. Instead, the DevOps team should be rewarded when the product, collectively, is excellent, and customer happiness is maintained (Walls, 2013).

The suggested characteristic of open communication segues very neatly into the characteristic of respect. Respect should be inherent in any team but is increasingly vital in those of DevOps culture. As discussed, DevOps teams are comprised of different people with diverse skills and mindsets. When a team contains distinct members and is driven by a culture where communication is essential, it is increasingly important to have respect for one another (Walls, 2013). It is not necessary for team members to like each other, but they need to respect and listen to everyone as well as have respectful

discussions and recognize the value of the contributions of diverse team members (Walls, 2013; Feijter et al., 2017).

Lastly, Walls (2013) express the importance of establishing trust within the team. Walls (2013) accentuates that within a DevOps culture, tools will not matter if there is distrust. Each team member, or function, of a DevOps team, needs to trust that everyone is doing what they can and in the best way to achieve success and not to introduce failure for specific people or functions; developers must trust that the QA team is not incentivized by sabotaging developers success.

Consequently, it is essential, when transitioning to or working within, a DevOps team to have a focus on the culture and communication through the various aspects presented above. For a DevOps team to be successful, there has to be an emphasis on creating a culture that promotes open communication, knowledge sharing, trust and respect, teamwork, and alignment of motivation and expectations of release and quality.

As culture is a detached term, it is challenging to quantify it and promote a best practice, since it is relying solely on the affected people and their mindsets. The challenge of quantifying culture is further backed by the lack of established knowledge within the area of culture in DevOps teams. The majority of the available literature agrees that culture and communication are crucial for a DevOps team, but only the research of Walls (2013) and Feijter et al. (2017; 2018) identify specific focus areas or characteristics that are increasingly essential for DevOps teams.

As such, there is no solid grounding and foundation for what the optimal DevOps culture is, and if it is plausible to determine what that culture is. As there is no solid theoretical foundation for what an optimal DevOps culture is, the measurement and assessment of such will be prone to an increased amount of subjectivity. Therefore, this thesis recognizes a lack of research within the field of quantifying and measuring culture in a DevOps organization.

2.5. DevOps Value

Most of the research suggests that successful adoption of DevOps in an organization will deliver value to that organization (Ebert, 2016; Elliot, 2014). As previously mentioned, the mere introduction of DevOps does not necessarily create value, and as such, most of the research establishes

prerequisites of culture and capabilities when discussing value generation through DevOps. Thus, when discussing value in this literature review, it presupposed that the organizational DevOps approach is a success. When discussing value generation in DevOps, it is essential to determine what value is to efficaciously discuss and compare the available theory and cases of DevOps. As mentioned, there is a consensus on the positive generation of value through DevOps in the available literature. However, there are few scholars that debate what exactly value is and what the term covers in the context of DevOps.

The preponderance of the reviewed literature proposes that the value gained from DevOps is organizational agility (König & Steffens, 2018; Ebert, 2016; Krancher, Luther, & Jost, 2018; Aiello & Sachs, 2016; Reed, 2014; Mohamed, 2016). However, the focus and assumption of what organizational agility is and what it incorporates differentiate between the scholars and cases. Some explain organizational agility as faster value delivery to the customer (Köning & Steffens, 2018); and some invoke the decrease of miscommunication and errors, and accelerated problem resolution, as the driver (Ebert, 2016). Some see the ability to integrate small amounts of code gradually to avoid future problems as agility (Aiello & Sachs, 2016), and yet others see some of the mentioned advantages, all of them, or something completely different as being organizational agility (Reed, 2014; Krancher et al., 2018).

As such, organizational agility can be seen as an umbrella term that incorporates some or all of the mentioned capabilities, and perhaps many more, suggesting that the assessment of agility in organizations is prone to an aspect of subjectivity. To simplify, the definition of organizational agility used in this thesis will be the one of Lu & Ramamurthy (2011): *"The ability to cope with rapid, relentless, and uncertain changes and thrive in an environment of continually and unpredictably changing opportunities"*. Due to the term of organizational agility reaching across a large area of capabilities, it needs to be fractured into tangible processes within the organization to understand the value drivers in DevOps organizations. Therefore, the aspect of value generation within the area of DevOps needs further examination to determine a more precise definition comprehensively.

Gill et al. (2017) and Virmani (2015) introduce several benefits tied to DevOps that increase the overall value and profitability of the concept. The most important of the value-driving benefits are enhanced internal communication, especially between the development and operation teams; a

reduction in human errors; a more streamlined and effective production pipeline from development to the customer; and the possibility of quick continuous customer feedback by providing software in an operations environment for use without unnecessary delays.

Dingsøyr & Lassenius (2016) enforce the aspect of faster customer feedback loops and general stakeholder empowerment, but also introduce a benefit comprised of organizational visibility. The organizational visibility is apparent through the fact that continuous development of software to market increases the visibility of the organization in the market, which can increase the customer base. Also, an organization can be recognized for its innovative and technological characteristics in using DevOps, which in turn can reflect positively on the organizational brand.

Wiedeman et al. (2019) present two levels of benefits that drive value: The organizational level and the team level. Within the organizational level are the rapid customer feedback and the increased speed of delivery on software features to customers that leads to enhanced customer satisfaction and profitability. The team level includes aspects such as the increased collaboration between employees and improved work-life balance of employees.

Kim (2018) introduce three main, more managerial value drivers of DevOps. Firstly, faster time-to-market through reduced cycle times and higher deploy rates. Secondly, increased quality in both availability and change success rate as well as fewer failures. Lastly, increased organizational effectiveness through increased time spent on value-adding activities, less waste, and increasing value delivery to customers.

As such, there are several value-adding activities of DevOps presented in the available literature. The capabilities presented through the works of Gill et al. (2017), Virmani (2015), Dingsøyr & Lassenius (2016), Wiedeman et al. (2019), and Kim (2018) are all arguably part of the organizational agility but deem more measurable and tangible.

Walls (2013) introduce several values that can measure some of the capabilities presented above. These are average time-to-market for new features, software deployment time, number of defects detected in testing before production release, and performance and user feedback. The values presented by Walls (2013) are arguably targeted upon measuring the organizational level benefits. In

contrast, the team-level benefits of collaboration and communication inter-organizationally, as well as the work-life balance of employees, can be extensively more challenging to determine the value of why the assessment of such is going to be more prone to the subjectivity of the researchers. Thus, there is an inconsistent definition of DevOps value in the existing literature. In this thesis, we view organizational agility as being the primary value derived from DevOps.

2.6. Software Maturity Models

IT systems enable organizations to improve their capabilities, processes, practices, structures, and knowledge, all of which have a direct impact on the competitiveness of an organization. Responsibility for effective and efficient use and design of IT systems depends on the organizational IT management (Becker, Knackstedt & Pöppelbuß, 2009). The main goal is to continually improve the organizational IT performance to establish IT excellence and, thus, enable a more productive and competitive business. However, continually improving and maturing the IT of organizations requires an assessment of the current state of the IT capabilities, systems, and services. This assessment entails an identification of the goals, external requirements, and benchmarks of the organization, as the level of IT maturity varies from business to business. Assessing the current state of organizational IT capabilities, systems, and services can be problematic as it is challenging to determine precisely what and how to measure, as well as what to compare the capabilities to, in order to assess the current state (Becker et al., 2009).

Maturity models are excellent tools for addressing these issues. They are among the most common theory in the field of improving organizational performance (Khoshgoftar & Osman, 2009). Maturity models are ideal for deriving helpful information used for prioritizing improvement measures as well as denoting the maturity progression. There are hundreds of different maturity models, each focusing on their specific area of expertise (Khoshgoftar & Osman, 2009). Despite the difference in fields of practice, a maturity model usually consists of a sequence of maturity levels, each representing a stage of requirements or objectives that the organization must achieve. The organization progresses from one level to the next by achieving the objectives defined in the specific maturity model. Maturity levels cannot or should not be skipped as each level provides a necessary foundation for the next level (Khoshgoftar & Osman, 2009).

Software process maturity models typically stem from either the initially prevalent Capability Maturity Model (CMM) or the evolved version, Capability Maturity Model Integrated (CMMI). The software process maturity models help to describe the underlying practices and principles for maturing a software process and, thus, help organizations achieve maturity through well-defined steps in order to increase the software process and quality (Ramanujan & Kesh, 2004).

The CMM model was initially introduced by The Software Engineering Institute (SEI) that was established by the American government to define software standards for the Department of Defense as well as seeking to improve their overall software quality. The initial purpose of the model was to serve as a control mechanism to ensure the quality and processes of a contractor (Ramanujan & Kesh, 2004). Despite laying the foundation for future maturity models, the CMM model is criticized for having several architectural flaws. The model utilizes an activity-based approach to measure the maturity of software processes. This approach leads organizations to an inadequate level of maturity as completing the predefined activities should, according to the model, advance the organization to the level of maturity. However, there is no way to quantify whether the activities were completed in the required manner, and thus, organizations may achieve a higher but false level of maturity (Royce, 2002). Several other models have later been built upon the groundwork of the CMM model, trying to incorporate solutions and disciplines to the critical challenges faced by the CMM. However, with the increasing number of models and their different focus areas, organizations that seek to implement process maturity models have difficulties selecting one that matches their needs. To overcome the inconsistencies, overlaps, and integration problems, the CMMI was developed as an initiative to integrate the many different process maturity models into one unified set (Royce, 2002).

The CMMI model introduced numerous improvements, with the two main parts being more flexible and result oriented. In terms of flexibility, the CMMI allows for a staged and more continuous method compared to the waterfall approach applied by the CMM. Furthermore, the movement from a sequential approach to an iterative lifecycle allows the CMMI to integrate the latest best practices from the concerned industry continually. Even though the CMMI uses a similar activity-based approach, the emphasis on the outcome and results are far more significant compared to the somewhat outdated CMM (Royce, 2002). With the model being result-oriented, organizations using the CMMI, rather than the CMM, will likely get a more applicable or actual impression of their software maturity. This will assist them in planning and advancing their processes as well as improving their software

quality. The CMMI model has laid the foundation for other maturity models to emerge within agile environments to assist organizations in assessing their maturity level (Royce, 2002; Ramanujan & Kesh, 2004).

2.7. DevOps Maturity Models

Reviewing the literature on DevOps shows that there are several maturity models related to the topic. However, based on Gasparaite & Ragaišis' (2019) research, only three of the existing DevOps maturity models should be considered for further investigation as they are deemed more comprehensive compared with other models. Zarour et al.'s (2019) research support this claim as their study identified seven DevOps maturity models, of which three are the same models that are considered relevant by Gasparaite & Ragaišis (2019). The models that were identified as adequate for future studies, and therefore included in this study, are The Hewlett Packard Enterprise Model, hereafter referred to as the Topham model, by Inbar et al. (2013); Samer I. Mohamed's (2015) maturity model; and the Focus Area Model by Fejter et al. (2017).

The comparison study by Zarour et al. (2019) showed that most DevOps maturity models follow either the CMM or the evolved CMMI. Inbar et al.'s (2013) Topham model (*Figure 3*) is subject to these findings as it is aligned with the standards of the CMMI model. The Topham model consists of five maturity levels and is designed to cover the complete lifecycle of an application or service. Inbar et al. (2013) have defined three dimensions as measure areas: Process, automation, and collaboration.

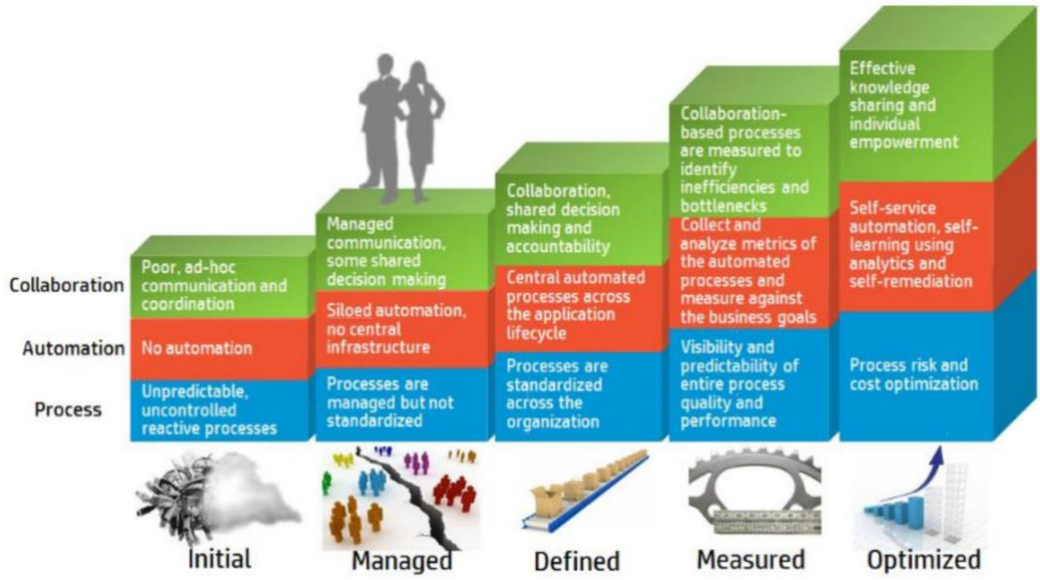


Figure 3: The Topham Model by Inbar et al. (2013)

The model was initially introduced to cover the full lifecycle of an application or service, but it lacks simplicity and specificity, according to Gasparaite & Ragaišis (2019). Organizations should, therefore, seek to utilize the Topham model as a guideline rather than to implement it as an organizational instrument for assessing the level of DevOps maturity. Furthermore, the Topham model does not contain a level 0, and therefore, it does not account for organizations that are yet to implement any form of DevOps mechanisms. In detail, all organizations are level 1 by default, indicating that they do have some knowledge and structures in place for the use of DevOps, which might not be the case (Zarour et al., 2019).

Mohamed's (2015) proposed maturity model (*Figure 4*) is an evolved version of the previously described Topham model. Mohamed's model has the same five levels of maturity, compared to the Topham model, but differs as it is measured against four dimensions: Communication/collaboration, automation, quality, and governance. Similarly, the purpose of the model is to cover the entire lifecycle of an application or service (Mohamed, 2015).

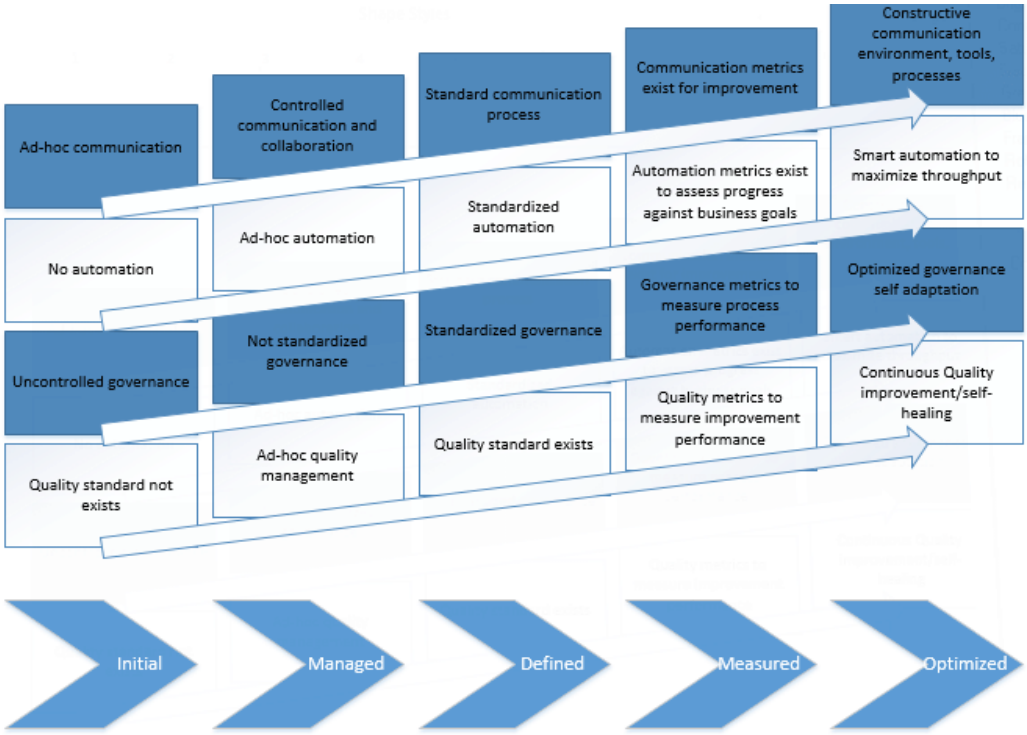


Figure 4: The maturity model by Mohamed (2015)

At the initial level, collaboration and communication are purely ad-hoc, and new deployments are only released when all parties feel they are ready, which leads to deployments being heavily reliant

on individual talent. As deployments are reliant on individual talent, it is challenging to predict whether a service or application is of production quality. Furthermore, the release process is dependent on numerous manual steps, as no automation is implemented in the process, which can force the release cycle to take multiple days or weeks. Additionally, the process is not governed and, therefore, perceived as uncontrolled. At this level of maturity, customers will often experience errors in functionality due to the number of manual steps in the process (Mohamed, 2015). The levels of maturity advance similarly to the Topham model, where every level is defined with features of incremental changes to each of the four dimensions (Zarour et al., 2019). At the final level of maturity, the collaboration is enriched with a constructive environment, tools, and processes. The process is issued with smart automation to maximize throughput, which allows the team to initiate failures to see if the system acts according to the defined actions, leading to a state of continuous improvement of quality. Organizations at this level of maturity can initiate business experiments to find new ways of delivering value to customers (Mohamed, 2015).

Although Mohamed's (2015) maturity model implements an additional dimension as well as seeking to be more result-oriented, it still lacks facets in the same areas as the Topham model. Organizations should seek to utilize the model as a guideline rather than try to implement it as an assessment tool. This is mainly due to the models presenting the approach to DevOps in a simplified manner, which means it can be challenging to apply the models in complex organizational situations.

Another approach to a DevOps maturity model is the Focus Area Model by Feijter et al. (2018). The model distinguishes itself from the traditional maturity model due to its focus area architecture. The Focus Area Model (*Figure 5*) is designed to enable a fine-grained maturity process for organizations, which is why the model contains ten levels of maturity compared to the traditional five (Feijter et al., 2017; 2018). The Focus Area Model is built on the foundation of a competence model that identifies three main perspectives: Culture and collaboration; product, process, and quality; and foundation.

Each perspective is comprised of different focus areas that all are deemed relevant for the DevOps maturity of an organization (Feijter et al., 2017). The focus areas are individually characterized by several capabilities that are represented as letters in the model. The capabilities are placed in the model corresponding to the level of maturity. For instance, the first defined capability within team organization is *A - separate teams*, whereas the last defined capability is *D – Cross-functional teams*

with knowledge overlap. The positioning of the capabilities is based on prior research as well as the dependencies among the capabilities. For example, there is a need for establishing some form of communication before any knowledge sharing can take place (Feijter et al., 2017; 2018). A detailed description of the 63 capabilities can be found in *Appendix B.*

Focus area \ Level	0	1	2	3	4	5	6	7	8	9	10
Culture and collaboration											
Communication		A				B	C			D	E
Knowledge sharing				A		B	C				D
Trust and respect							A	B	C		
Team organization		A	B						C	D	
Release alignment				A					B	C	
Product, Process and Quality											
Release heartbeat		A				B	C		D	E	F
Branch and merge			A	B		C		D			
Build automation			A	B		C					
Development quality improvement			A			B		C	D	E	
Test automation				A	B	C			D		E
Deployment automation					A	B		C			D
Release for production					A			B	C	D	
Incident handling			A					B	C	D	
Foundation											
Configuration management			A	B		C					
Architecture alignment			A					B			
Infrastructure				A			B	C	D		

Figure 5: The Focus Area Model by Feijter et al. (2018)

The Focus Area Model is, by far, the largest and most comprehensive DevOps maturity model available. Unlike the two other models, the Focus Area Model tries to identify which parts of the software process, and in which order, organizations should seek to focus on to mature their DevOps approach. However, the model is based on data derived from one organization; hence the model might be prone to bias, which impacts the generalization of the results (Feijter et al., 2017). Furthermore, the focus areas, as well as the capabilities, are yet to be validated by DevOps experts, and thus, the applicability of the maturity model could be questioned (Gasparaite & Ragaišis, 2019).

Generally, the available DevOps maturity models all approach the concept differently but somewhat align in terms of the capabilities described in each model. For instance, communication or deployment automation seems to be rather identically perceived by the different models. However, the models differ in their definition of what is required to reach the highest level of maturity. Also, the individual, organizational needs are not accounted for in the models, as the level of maturity might not bring the same marginal value for every organization (Gasparaite & Ragaišis, 2019).

Research on DevOps maturity models is scarce, indicating the general lack of empirical studies that document and validate the adoption of DevOps maturity models. Another concern regarding the DevOps maturity models is the lack of assessment methods for determining or estimating the current level of maturity (Zarour et al., 2019).

3 Methodology

This section presents the methodological considerations and choices we have made in this thesis. We elaborate on the adopted research philosophy, research approach, and research design, and the longitudinal aspect. Also, we describe the utilized data as well as the empirical methods used for data collection and processing.

3.1. Research Philosophy

This thesis utilized the pragmatic paradigm in the investigation of DevOps in the two case organizations. The goal of this thesis is to investigate how organizations can mature their DevOps approach. As we aim to contribute knowledge to a reasonably unknown subject, we deem pragmatism as a good fit due to the acceptance of multiple realities existing in empirical inquiries (Creswell & Clark, 2011).

In the pragmatic paradigm, there is a focus on the research question and central problem, how the problem is solved, and what the cause of the problem is. Furthermore, pragmatism brushes aside the division of qualitative and quantitative research by attending the researchers' focus on the investigation of a particular concept (Feilzer, 2010). Thus, the relatively open boundaries of pragmatism allowed us to select methods solely based on their suitability for addressing the research question.

In this thesis, we further seek to produce knowledge from the investigation of the two case organizations that apply to other settings under varying practical circumstances. The pragmatic paradigm offers epistemological justification for combining multiple sources of knowledge to find workable solutions and gain an understanding of people and the world in which we reside and practice (Nowell, 2015). Moreover, the primary goal of pragmatic knowledge creation is to produce knowledge, to which controlled improvements or changes of human existence can be made (Goldkuhl, 2012). As part of the investigation, workflows, culture, challenges, and level of maturity within DevOps are examined in the case organizations, resulting in a discussion of findings that could deem practically relevant and useful for the case organizations or other organizations looking to adopt or mature DevOps.

Thus, we believed that a different paradigm could not have achieved the same desired result. For instance, the use of positivism could have generated entirely different results, since positivism aims to produce knowledge that can be generalized to a large scale. Positivists risk neglecting softer elements such as individuals' understanding and perceptions of critical events or issues, which would have left significant gaps in essential knowledge of the case organizations (Lan, 2018). If this thesis had focused upon creating knowledge that could be directly replicated in other settings, such as creating a general framework, a positivistic approach would have been more applicable. Another alternative is the interpretive paradigm that, for years, has been an established and adapted paradigm for qualitative research. However, interpretivism seeks merely to observe the world, whereas the intention of this thesis is not only to observe but to intervene and to generate knowledge for action and change (Goldkuhl, 2012).

Goldkuhl (2012) emphasizes a need for a larger paradigmatic consciousness, as qualitative research of information systems frequently couples pragmatism and interpretivism, but often implicitly. As such, we follow this observation, as we see the coupling between pragmatism and interpretivism as necessary due to "*the difficulties in reducing the complex social and technical phenomena in IS-fields"* (Goldkuhl, 2012). Therefore, the coupling of the two paradigms within this thesis is made consciously and explicitly due to the complex construct of the DevOps concept that involves both social and technical aspects.

3.2. Research Approach

In practice, it is often difficult, when working with social science projects – especially case studies – to separate inductive and deductive approaches, due to the two research approaches being woven into each other and occurring concurrently throughout the entire research process (Andersen, 2014).

The pragmatic research philosophy allows us, unlike the positivistic and interpretive research philosophies, to integrate more than one research approach within the same study (Nowell, 2015). As this thesis sought to investigate a specific situation in two case organizations through a theoretical lens of maturity models and previous literature on DevOps, we utilized deductive reasoning. Thus, the initial processes of creating a knowledge foundation, shaping the right research question, and constructing the interview guide through operationalization as well as the initial data handling of coding, analysis, and interpretation were conducted through deductive reasoning. However, in the

later analysis and interpretation of the collected data, it was appropriate to deliberate on how valid and generalizable the contexts and findings are. When raising these questions of whether the results from the investigation can apply to other organizations, we introduced inductive reasoning. Thus, we utilize both research approaches to address the research question of this thesis comprehensively.

3.3. Research Design

A research design is a designation for the approach to how the concept that is the subject of the investigation is explored. In more detail, the research design represents the combination of techniques that are utilized in the collection, analysis, and interpretation of data. The purpose of choosing the right research design is to ensure that the obtained data and documentation are capable of, as unequivocally as possible, to guarantee a sufficient answering of the research question (Andersen, 2014). As such, the chosen techniques of analysis and interpretation must reflect and support the means of the data collection and the posed research question.

As this thesis sought to investigate the presence and maturity of DevOps in two different organizations, the research method was conducted as a paired-case study. According to Yin (2002), the approach of case studies is very widely utilized within all areas of social science, such as sociology, history, and economy, and even more widely used in academic projects and theses within the area of social science. The paired-case study method enables the researchers to investigate an extended range of variables in two settings or organizations, to produce reliable explanations or depictions of the investigated topic, specific to the investigated organizations.

According to Kruuse (2007), there is a tendency to consider case studies as being qualitative investigations. It is not meant to discard any case studies that are based on quantitative data or include quantitative aspects, but the majority of research with a case study method is typical of qualitative characteristics. The presence of qualitative characteristics is no exception in this thesis, as we derived the data that laid the foundation for the investigation from interviews. The qualitative data included is of both primary and secondary character as the research relies on data collected from different sources, times, and people. We further elaborate on the origin and specifications of the data in section *3.5 Data Collection*.

There were two main reasons for the choice of conducting interviews and basing the thesis on qualitative data. The first reason was due to the topic of investigation and the approach the research question proposes. As the majority of literature on DevOps highlights the importance of culture, it requests the investigation of softer elements that are not bound in hard data, such as culture and collaboration. When investigating elements that are bound in human behavior and interaction, it can be complicated to capture the entirety of something so complex using quantitative data, compared to qualitative data, and especially interviews, where respondents can explain and elaborate on their answers (Kruuse, 2007).

The second reason behind the choice of using interviews, and thereby qualitative data, was the dependency on the secondary data this thesis utilizes. The secondary data of this thesis stems from interviews conducted in the two case organizations in 2015. As the previous data originates from semi-structured interviews, it was appropriate to utilize the same or a similar data collection technique to ensure consistency and increase comparability between the data points. However, this is not to reject the possibility of using different data collection techniques in a longitudinal study. To ensure the comparability of the data and increase the validity of the results, it is advocated to utilize consistent methods when collecting data (Holland, Thomson, and Henderson, 2006).

When conducting interviews to investigate a selected topic, it is essential to consider the desired number and type of respondents. This further raises questions on how to choose the right respondents that can contribute to the most valuable data to encompass all aspects of the investigated topic fully. The desire is to say something about all elements, in this case, the organizations and DevOps teams, by choosing certain people or teams to investigate, which introduces the term of *inference*. Inference means to predict something about the total population, based on a subpopulation (Andersen, 2014). The reasoning behind introducing inference is due to the considerable amount of resources it would take to investigate every single person connected to DevOps in the two organizations. As such, we only chose a certain number of respondents from both organizations. However, an important aspect here was to ensure the presence of at least one respondent from each affected department, group, or team so that the least amount of valuable data was left uncovered.

As this thesis utilized the paired-case study method, it is defined as an intensive study due to the examination of few cases in order to understand causes and effects in-depth rather than as extensive

research using a large number of cases to determine commonalities in a population (Andersen, 2014). Examples of intensive research are often seen in case studies that investigate few survey units (organizations) in a vast amount of areas. Compared to an extensive study, such as a public opinion polling that investigates many survey units (people) in a relatively small amount of variables, such as their political stance. We have chosen to conduct an intensive study and focus solely on ProActive and Topdanmark, due to the numerous aspects included in the concept of DevOps that needs examination.

3.4. Longitudinal Analysis

This thesis utilizes a qualitative longitudinal paired-case study as the research design in the investigation of DevOps in ProActive and Topdanmark. This specific research design has allowed for a thorough examination of the DevOps concept over five years in the two case organizations.

Qualitative research is particularly appropriate for examining processes through its attention to context and particularities, which allowed us to research the individual cultures and processes in each organization. Qualitative *longitudinal* research is predicated on the investigation and interpretation of change over time and process in social contexts (Holland et al., 2006). As this thesis utilized secondary data collected from the case organizations in 2015, the longitudinal aspect was introduced. The longitudinal element has allowed for an investigation of causal relationships in (social) processes and change over time (Holland et al., 2006). With the combination of qualitative and longitudinal research, we adopted a research design that aligned with the use of maturity models and process theory to facilitate the answering of the research question.

According to Holland et al. (2006), the vast majority of longitudinal research is conducted using quantitative methods and data. The rationale behind a preponderance of quantitative longitudinal research is that qualitative longitudinal research is often affected by funding pressures and a lack of time among the researchers. As we used secondary data that was already collected, we did not encounter the mentioned challenges that qualitative longitudinal research is usually subject to.

If this thesis did not include a longitudinal aspect, it would not have been possible to investigate the case organizations to the same extent. The absence of a longitudinal element would have profoundly impacted the investigation of organizational processes over time. Additionally, it would have limited

the use of process theory, and visual maps, to identify causal relationships and possible patterns of events within organizational adoption and use of DevOps.

3.5. Data Collection

This section describes the empirical methods we have utilized by detailing the involved data, research methods, and techniques as well as their rationality towards this research. In our research, we solely applied qualitative data but introduced data triangulation, as the applied data stems from different sources, times, and people (Flick, 2004). We elaborate on the empirical methods used for data collection, data processing, and data analysis were directly associated with the research design, and a clear understanding of the different selected methods and their connection to the research design.

3.5.1. Operationalization

Operationalization is a central activity within the analytical realization process, and was used in this thesis to translate theoretical concepts into empirically measurable entities (Andersen, 2014). The purpose of the operationalization activity was to ensure that the collected data was accurate and could be used for further analysis.

Theoretical concepts are typically multidimensional and can be complex to break down into individual variables, why an essential activity in the operationalization is to identify all the variables associated with a theoretical concept. The identified variables can afterward be selected concerning their relevance to the research scope (Andersen, 2014). Besides, it can be difficult to measure variables that directly reflect the theoretical concept. For instance, theoretical concepts such as culture and mindset are complicated to measure and require additional attention. Such theoretical concepts require disintegration to replacement variables, or the summation of a set of multiple small empirical entities, to yield a useful result (Andersen, 2014).

We performed the operationalization activity prior to the data collection to establish a link between the theoretical foundation and the empirical elements. The theoretical foundation consists solely of the knowledge base established in the literature review. Based on the knowledge base, theoretical concepts were derived and broken down into empirical variables. We designed detailed questions for the qualitative interviews with the variables as the foundation. Please refer to *Table 2* for the complete operationalization table.

Table 2: Overview of operationalization

Theory	Opertionalization	Question
Culture	Communication	- How do you commincate within the team? - Do you use any special methods or communication channels (mail, chat, etc.)?
	Knowledge Sharing	- Has there been made any effort to include everybody in every team member's process? - Are every team member included in the entire process from development to deployment?
	Collaboration	- How do you cooperate in the team?
	Team organization	- When the team was established did you notice any differences in culture between the teams that were put together? - Was there done any effort in bringing down the formal wall between the entities?
	Trust & Respect	- Do you feel that all team members have respect for and listen to one and other? - Do you feel that all team members have respect for one and other's role descriptions? - Is there a sense of a common goal within the team?
Process	Release	- What is the process from a feature in development to production? - How do you decide what features to develop for customers? - How often do you deploy new features? - How long time does it take for a finished feature to reach the customer?
	Monitoring	- Do you monitor your software/product?
	Automation	- How much of your process from development to production is automated (integration, test, deployment)?
	Technical infrastructure	- How is the underlying infrastructure of your product?
	Tools	- Is there an alignment between the tools you use in the team?
	Incident handling	- If something goes wrong in production how do you handle it? - Do you experience less erros after adopting DevOps?
Challenges	Organizal structure and size	- Do you feel like your team size or organizational structure has had any effect positive or negative on your apporach to DevOps?
	Obstacles	- Have you encoutered any large problems within the team? - Is that something that hinders you from progressing with DevOps?
Value	General value	- Do you feel that the transition to DevOps has provided any value? And is there more to gain?
	Time to market	- Has your time to market been reduced after the introduction to DevOps? - Do you see it as an advantage that you can deploy more often than before?
	Organizational brand	- Do you feel that the introduction of DevOps have improved your organizational branding (more technological/innovative brand)?
	Customer feedback and satisfaction	- Has your customers been more involved after the introduction of DevOps? - Has your customer satisfaction increased?

3.5.2. Primary Data

In this thesis, we solely used qualitative data as the study investigated aspects that were difficult to measure and quantify, including aspects such as company culture and employee mindset. The primary data provided a contemporary image of the two case companies' adoption and approach to the DevOps concept. Weeks before the primary interviews, we conducted two informal, unstructured interviews with contacts from each case organization. As part of these interviews, we defined the scope of the research and agreed upon access to resources (i.e., selected respondents from each case organization). In collaboration with the two organizations, we based the selection of respondents on the individual respondent's role and involvement with DevOps.

We collected the primary data through semi-structured interviews held with the employees from the two investigated case organizations. The data collection technique of semi-structured interviews was selected as it provided a relatively open setting, which helps the respondents to perceive the interview as a conversation rather than an interrogation (Andersen, 2014). We chose the semi-structured technique as opposed to the structured technique, as it acknowledges that meanings of words and extent of vocabulary can vary among respondents, which we deemed essential as DevOps can be a complex concept. Furthermore, the data collection technique allowed the respondents to delve and emphasize points that they felt were the most relevant (Barriball & White, 1994).

The flexibility of the semi-structured interview is an advantage when conducting research, where language barriers exist, as the interviewer can select exact words that ensure the validity and reliability of the data, which is the case of this thesis, as non-native English speakers are part of the respondents (Barriball & White, 1994). Furthermore, the flexibility of the technique allowed us to exceed the constructed interview guide and ask additional questions regarding relevant areas that could appear during the interview or to acquire information about uncovered topics (Andersen, 2014). Please refer to *Table 3* for an overview of the interviews and the role of respondents.

Table 3: Overview of respondents (primary data)

Respondent number	Organization	Role/Title	Approximate duration (min)
1	ProActive	Developer	60
2	ProActive	Senior Developer	40
3	ProActive	Tester	30
4	ProActive	Technical Product Delivery Manager	60
5	ProActive	Technical Product Delivery Manager (retrospective interview)	40
6	Topdanmark	Developer	30
7	Topdanmark	Developer	40
8	Topdanmark	Product Owner	50
9	Topdanmark	Developer (Hawks)	40
10	Topdanmark	IT Development Manager and Architect	60
11	Topdanmark	Scrum Master	30

12	Topdanmark	DevOps Specialist (IT Operations)	30
13	Topdanmark	IT Development Manager and Architect (retrospective interview)	70

The developed interview guide consists of the questions extracted from the operationalization activity. However, questions related to the history of the organizations were only included in interviews with long-tenured employees. The interview guide served as a guiding structure in the interviews. However, we made attempts to continuously adjust the questions to be as relevant as possible to the individual respondent. Furthermore, we deliberately chose to include a minimum of theoretical terminology in the questions to avoid any confusion among the respondents. *Figure 6* shows a snippet of the interview guide (please refer to *Appendix C* for the full-sized version).

Area of Concern	Question
Introduction	• What is your current job description? • How long have you been at this company? • Can you describe your tasks and responsibilities?
The Company	• How would you describe the company culture? • What is the organizational IT structure?

The Teams	• Where are the different teams located?
	• How do you communicate within the team?
	  o Do you use any special methods or communication channels (mail, chat, etc.)?
	  o Do you use any specific knowledge sharing platform?
	• When the team was established did you notice any differences in culture between the teams that were put together?
	  o Was there done any effort in bringing down the formal wall between the entities?
	  o Do you feel that all team members have respect for and listen to one and other?
	• Is there a sense of a common goal within the team?
	  o Are the teams rewarded in any specific way? What are they measured on?

Figure 6: Snippet of the interview guide

The interview agenda and further information about the topic were not distributed to the respondents before the interviews, as it was made clear at the preliminary interviews that the respondents had either been introduced internally to DevOps or worked with it daily. We initially scheduled the interviews to be conducted at the headquarters of the two case companies. However, due to the outbreak of the COVID-19 virus forcing all Danish companies to close office spaces, only four of the scheduled interviews were held in person. Consequently, we conducted the majority of the interviews as virtual meetings through Microsoft Teams. The interviews were in either English or Danish based on the preference of the individual respondent. The duration of the interviews ranged between 30-60 minutes and were all audio-recorded and subsequently transcribed.

Two retrospective interviews were successively held with employees from the case organizations to obtain additional insight into their DevOps approach. The two respondents in the retrospective interviews were selected based on their tenure at the organizations, as the need for data on the exact occurrence of specific events, decisions, and activities was necessary for further analysis. We encouraged the respondents to research the events before the interview and kept an informal atmosphere to provide the respondents with time to recollect. The retrospective interviews were conducted as unstructured interviews using Microsoft Teams and were likewise audio-recorded and subsequently transcribed.

3.5.3. Secondary Data

Due to the applied longitudinal research design of this research, the necessity for data from other time points was a requirement. Therefore, we included secondary data from previous research conducted by Nielsen, Winkler, & Nørbjerg (2017) in this research. The interviews from their research were likewise from ProActive and Topdanmark. They collected the data in 2015 from 15 role-specific interviews (five at Proactive, ten at Topdanmark) with different stakeholders (please refer to *Table 4* for an overview of the interviews and role of respondents).

Table 4: Overview of respondents (secondary data) (Nielsen et al., 2017)

Respondent number	Organization	Role/Title	Approximate duration (min)
14	ProActive	Director Solutions	N/A
15	ProActive	Product Owner	N/A
16	ProActive	Developer/Solutions Architect	N/A
17	ProActive	Tester	N/A
18	ProActive	IT Professional (IT Operations)	N/A
19	ProActive	Director Solutions (retrospective interview)	N/A
20	Topdanmark	Service Owner	N/A
21	Topdanmark	Specialist	N/A
22	Topdanmark	Product Owner	N/A
23	Topdanmark	Application and Architecture Responsible	N/A

24	Topdanmark	Developer	N/A
25	Topdanmark	Service Owner, Specialist, Release Manager (IT Operations)	N/A
26	Topdanmark	Service Owner	N/A

Secondary data is defined as data that was initially collected for a different purpose but reused to explore new research areas (Andersen, 2014). Nielsen et al.'s (2017) research paper investigated how companies or teams adopting DevOps could assess their fulfillment of essential DevOps elements. While their research differs in scope, it provided valuable insights and reusable data for assessing the organizations' previous level of DevOps maturity. Nielsen et al. (2017) utilized the semi-structured interview technique for collecting their data, which is the same technique applied in this thesis for the collection of the primary data.

3.5.4. Referencing Interview Quotes

Most of the interviews are in Danish (both from the primary and secondary data). Accordingly, we translated all quotes cited in the thesis to English. We have translated all quotes to the best of our ability while trying to maintain the original meaning and emotions of the individual respondent.

3.6. Data Analysis

The interviews resulted in a rather large and comprehensive set of transcribes. We coded the transcribes to apply structure to the data through categorization, where responses were connected to the presented theory. The coding process was conducted on both the primary and secondary data to ease and rationalize the further handling and interpretation of the data.

The coding process included two stages of coding. Firstly, the transcribes were coded according to relevance for the analysis and the research question. In this stage, we coded the transcribes to correspond with three levels of relevance: Relevant, potentially relevant, and irrelevant. Of these, the two categories, relevant and potentially relevant, were included in the further coding. The parts coded as irrelevant in the transcribes were mainly small talk and answers of respondents that were not relevant to DevOps and therefore excluded from further analysis.

The different fragments were color-coded in the following way: relevant fragments were highlighted with a green color, potentially relevant fragments were highlighted with an orange color, and irrelevant fragments were not highlighted. After we conducted the first stage of coding, the transcribes were fragmented into smaller, more relevant parts, where all irrelevant data was excluded from further analysis.

In the second stage of the coding process, we coded the relevant and potentially relevant fragments to chosen concepts presented in the theoretical foundation of the thesis. In this stage, we coded the data fragments to address one or more of the following:

- Culture and collaboration
- Product, process, and quality
- Foundation (technical infrastructure)
- DevOps Value
- Challenges
- Time (specific mentions of timeframe)

We based the selected theoretical concepts for the second coding stage on their pertinency towards the research question. Culture and collaboration; product, process, and quality; and foundation directly relates to the maturity model used for assessing the drivers and capabilities for progressing DevOps maturity. To determine what hinders organizations from maturing their DevOps approach, we coded the fragments to challenges related to the two organizations' DevOps approaches. By coding fragments to DevOps value revealed areas that generated value for the two organizations. Lastly, coding fragments according to time assisted in the creation of visual maps. By coding the relevant fragments of the transcribes to their connection with the chosen concepts, the data set became more tangible for the later interpretation and analysis.

4 Case Descriptions

4.1. ProActive

ProActive is a Danish consultancy firm that assists private and public organizations with their digital transformations (ProActive, 2020). ProActive is one of the leading Microsoft partners in Scandinavia and is specialized in the technologies developed by Microsoft. In short, they describe that their main activity is to "*help make it easy, clear, and advantageous to use modern technology*" (ProActive, 2020). The headquarters of ProActive is in Copenhagen, but they have smaller offices located in Odense, Aarhus, and Aalborg. ProActive was founded in 1997 and currently employs approximately 250 people (ProActive, 2020).

Despite being a well-established consultancy firm, ProActive also develops a software product, IntraActive, which is an intranet product based on Microsoft's platforms, SharePoint Online and Office 365. IntraActive is a traditional intranet that serves as an internal platform providing the users with access to essential tools and documents while also affording enhanced internal communication. The development of the product started in 2013 as an intranet project for a customer. However, due to the growing demand for a standardized intranet product, ProActive decided to form a dedicated intranet team in 2014. IntraActive is sold to multiple companies covering a broad range of industries within both the public and private sectors. The majority of the customers are located in Denmark, with few exceptions in Sweden, Brazil, and Germany (IntraActive, 2020).

The team behind IntraActive consists of 15 employees, of which 13 are at the headquarters in Copenhagen. The two remaining employees are based in Brazil and are contracted through a Brazilian software company. The team is split into two smaller groups: the business team and the technical team. The business team includes the product owners, marketing responsible, and the team director. The technical team consists of all the developers, one dedicated operations employee, and a product manager (Respondent 1, ProActive, 2020).

IntraActive was initially delivered as a typical software where the customer bought the product, and the ownership was transferred. The product was updated through a quarterly release cycle, but customers had to buy the new releases separately as this was not included in the product. All updates were manually installed and could take several days to perform as customer solutions typically

included multiple customizations that were not compatible with new versions. However, due to changing demands regarding software products as well as time-consuming development inflicted with many manual steps, the IntraActive team decided to change their delivery model. ProActive transitioned from a traditional delivery model to a Software as a Service (SaaS) delivery model in 2019. With the change to SaaS, ProActive introduced a continuous delivery setup within the IntraActive team. The new SaaS delivery model brought multiple benefits, but also specific requirements, for the internal release setup as well as the team culture (Respondent 5, ProActive, 2020).

4.2. Topdanmark

Topdanmark is a Danish insurance company that offers a wide range of insurance packages and financial services to both private and commercial customers. Topdanmark was founded in 1899 and is today the second-largest insurance company in Denmark. They currently employ 2400 people spread across the country, with their headquarters being in Ballerup (Topdanmark, 2020).

To support the business, Topdanmark employs an extensive IT department consisting of 400 employees. The IT department is divided into three divisions: development, operations, and a DevOps team. Development consists of several small teams that built web applications for Topdanmark's platforms. Connected to each development team are a product owner and a scrum master that oversee facilitating the development process. The operations teams handle the traditional operation tasks of monitoring software and hardware, incident handling, and configuration, and provisioning of technical environments. The DevOps team is in charge of building pipelines, version control, selecting and distributing tools, and other tasks associated with Topdanmark's delivery model (Respondent 10, Topdanmark, 2020).

All of Topdanmark's applications are built in-house. This decision originates from an IT project in the early 2000s, where Topdanmark needed a new integration layer. However, as no supplier could deliver the functionality needed, they decided to develop it themselves (Respondent 10, Topdanmark, 2020). This philosophy has, over the years, led Topdanmark to build large systems, dependent on its mainframe, that are specifically tailored to their business needs. Topdanmark's mainframe has been active for multiple decades. Thus, the old mainframe is not compatible with today's DevOps

technologies, which is why Topdanmark three years ago started a transition to becoming a more cloud-based company.

The management and employees of Topdanmark have a desire to transition away from the mainframe, but it has not been possible due to technical and economic reasons. Because of the dependencies of many systems on the mainframe, and the desire to become more cloud-based, Topdanmark's software development has been divided into two distinct processes with individual teams and technical architectures: The *Top-Up process* and the *Continuous Delivery process* (*CD-process).*

The Top-Up process contains several of Topdanmark's applications, including the CRM system of the organization. All applications within the Top-Up process are somewhat dependent on the mainframe. Due to the dependencies on the mainframe, Topdanmark cannot adopt a continuous delivery model to the Top-Up process. The Top-Up process has scheduled deployments approximately 10-12 times per year (Respondent 10, Topdanmark, 2020). The CD-process contains Topdanmark's newer applications and most frontend solutions. Servers and applications within the CD-process are hosted by Amazon Web Services (AWS). The "*ICE-WEB"* team is responsible for all newer applications and is the only team developing to the CD-process. The introduction of the AWS setup allowed Topdanmark to implement continuous delivery and thus provided them the ability to release software daily. However, due to the connections to the mainframe, a large number of complete applications are stuck in a limbo between test and production, waiting for the next release cycle of the mainframe (Respondent 10, Topdanmark, 2020).

5 Analytical Framework

This section will elaborate on the theories and models applied for analyzing and interpreting the acquired data. Despite advancing on the knowledge gained from the literature review, we adopt a maturity model for assessing the previous and current level of maturity of the two case organizations in the analysis. For the inclusion of the temporal aspect, we utilize process theory in the form of visual maps to visualize the sequence of DevOps related events performed in each organization during the last five years.

5.1. Maturity Assessment

We opted to use the Focus Area Model by Feijter et al. (2018) for assessing the maturity of the two organizations as we deemed it the most reliant, due to the scale and comprehensiveness of the model. In contrast to the other maturity models (Mohamed, 2015; Inbar et al., 2013), the Focus Area Model provides a set of measurable capabilities, which ease the maturity assessment task of the two organizations. Each area defined in the model possesses a set of capabilities that define the level of maturity of the respective area. However, worth noting is that the model lacks clarity in some areas. For instance, the level of maturity in terms of configuration management is not defined beyond level five as the highest-rated capability is placed at this level (please refer to *Figure 5*). The lack of defined levels applies to multiple areas in the model and is also present in between levels. For example, there are three undefined levels between capability A and B in the communication area. Thus, the model introduces a certain amount of subjectivity as we are required to estimate whether the investigated organization is in between levels or has ascended beyond the defined capabilities.

Another limitation of the Focus Area Model is the absence of certain concepts that are relevant to an organization's adoption and maturity of DevOps. For example, the model does not include aspects such as the employees' mindset and perception of DevOps. However, these types of softer concepts are a general lack of DevOps maturity models due to the challenging measurability of these. We are aware of this shortcoming and, therefore, try to incorporate these continuously throughout the analysis to get a more unobstructed view of the organizations' DevOps approaches. Furthermore, we recognize that assessing any focus area to level 10 reflects that the organization cannot improve within that area. However, we assess all focus areas according to the capabilities provided by Feijter et al.

(2018), and the assessment is therefore not a completely accurate depiction of the possible future levels of maturity due to the rapidly changing possibilities and environment of DevOps.

In section *6 Within-Case Analysis*, focus areas, capabilities, and levels will be referenced in the text with the use of *italics.*

5.2. Process Theory

When conducting dynamic studies, variance and process theory tend to emerge. Both theories put their emphasis on events, activities, and decisions. However, variance theory focuses on the relationship between the dependent and independent events as the theory seeks to provide explanations for outcomes and causes. Complementary to variance theory, process theories are concerned with the sequence and development of events that lead to an outcome. Thus, it is essential to understand patterns in events to develop process theories (Langley, 1999).

According to Langley (1999), process data collected from organizational contexts is characterized by several traits that make the data challenging to analyze. This is primarily due to the temporal characteristics of the data that can influence the precision, duration, and relevance of the data. Furthermore, the data deals with sequences of events where the background trends that shape a specific event are not included; hence the researchers have a difficult time grasping the underlying aspects. Process data also typically involves multiple units of analysis, making the data ambiguous and complex, as it can be challenging to determine which data to isolate as the reason for a specific event or decision (Langley, 1999).

The nature of process data is complex and ambiguous; hence it is paramount that an analysis strategy is adopted that helps facilitate the move from a multifaceted data base to a clear theoretical understanding. Langley (1999) proposes seven generic strategies for the sensemaking of process data. The strategies should not be regarded as step-by-step guides but rather as generic approaches to ensure theoretical understanding. The strategies can be used in combinations for breaking down the complicated data base. Despite presenting seven different sensemaking strategies, we only adopt one strategy for the analysis in this research: Visual mapping strategy.

Process data analysis may involve the manipulation of words, numbers, or matrices and graphical forms (Langley, 1999). According to Langley (1999):

> *"visual graphical representations are particularly attractive for the analysis of process data because they allow for simultaneous representation of a large number of dimensions, and they can easily be used to show precedence, parallel processes, and the passage of time."*

The use of visual maps to analyze process data is an attractive approach, as it allows for the illustration of several issue domains and parallel events as well as providing a clear indication of the temporal perspective. It is, however, important to highlight that visual maps serve as an intermediary step between the raw data and a more robust understanding of the concepts that are analyzed. As such, it is essential to move the visual map from a descriptive representation of process data to a taxonomically higher level (Langley, 1999). This is done through a comparison of multiple process maps for finding common sequences of events and activities that can facilitate a broader understanding of DevOps.

As such, in this thesis, we will utilize visual mapping to display the collected process data. The visual mapping strategy will allow for a large quantity of process data to be displayed in relatively little space. Additionally, the visual mapping strategy can be a useful tool in developing theoretical ideas (Langley, 1999), which further backends the choice of strategy, since the intention is to analyze the process data to discover patterns concerning DevOps.

6 Within-Case Analysis

6.1. ProActive Maturity Assessment

In this section, we assess the maturity level of ProActive for two states in time: the previous state (2015) and the current state (2020). The assessment of the previous state will serve as a reference point, and thus the assessment of this state of maturity will be more narrow. The assessment of the current state will be more in-depth to provide an insight into the current development cycle, culture, and technical infrastructure of the company. All focus areas of the Focus Area Model are included in the maturity assessment, but will not all be explicitly described, due to the large number of focus areas included in the model.

6.1.1. Previous Level of Maturity

The organizational structure of the IT department in ProActive was in 2015 split into the two traditional groups: development and operations. The teams were separated by location, and their communication mainly took place through email, documents, or quarterly meetings associated with an upcoming software release. This traditional separation meant that each team had different objectives and tasks, resulting in a lack of alignment. Especially the transfer of knowledge between the teams is observed to cause many problems. One employee describes the teamwork between the two teams: *"It is a challenge to make this continuous transfer of knowledge all the time because things happen relatively quickly in the product, but operations are always a bit behind"* (Respondent 14, ProActive, 2015). We, therefore, determine the *knowledge sharing* of ProActive to *level 5 (B)*, as there was a presence of knowledge sharing, but the knowledge was not shared actively. The *communication* is rated *level 6 (C)*, due to the presence of direct communication among employees. *Figure 7* shows the complete assessment of ProActive's previous level of maturity.

Firstly, it was not always the same people from the operations team that was responsible for the release and installation of new versions, which meant that additional time had to be spent on introducing the process. Secondly, the operations team was also responsible for handling other tasks, which resulted in installations of new product versions dragging out to a point where new releases would exceed ones that were yet to be installed on the customers' environments. As such, we rate the *release alignment* as *level 3 (A)*, as there was a shared understanding of releases, but no internal release heartbeat was present.

Focus Area / Level	0	1	2	3	4	5	6	7	8	9	10
Culture and Collaboration											
Communication		A				B	C			D	E
Knowledge sharing				A		B	C				D
Trust and respect							A	B	C		
Team organization		A	B						C	D	
Release alignment				A					B	C	
Product, Process and Quality											
Release heartbeat		A				B	C		D	E	F
Branch and merge			A	B		C		D			
Build automation			A	B		C					
Development quality improvement			A			B		C	D	E	
Test automation				A	B	C			D		E
Deployment automation					A	B		C			D
Release for production					A			B	C	D	
Incident handling			A					B	C	D	
Foundation											
Configuration management			A	B		C					
Architecture alignment			A					B			
Infrastructure				A			B	C	D		

Achieved level of maturity | Unachieved level of maturity | A-F Capabilities

Figure 7: Assessment of ProActive's previous level of maturity (2015)

ProActive previously delivered its product through a traditional release cycle, where a new version was released every four months. The release of new versions was a partly automated process that required a consultant from the operations team to manually run deployment scripts to install a new version on a client's environment. Each customer required a unique installation, and these could take multiple days, due to system dependencies, such as a SharePoint *crawl* (i.e., a crawl that is executed by Microsoft to gather data on SharePoint sites and update accordingly) that the organization had no control over (Respondent 18, ProActive, 2015). As ProActive had a fixed release heartbeat, releasing every four months, we rate the *release heartbeat* at *level 5 (B)*.

In terms of the *build automation*, the team ran automated builds every night for internal test environments *(level 3 - B)*. The automated build process was issued with *broken build detection* and *gated check-ins*. The development team also utilized traditional *manual code quality monitoring* methods, such as pair programming and code reviews (Respondent 16, ProActive, 2015). As such, we evaluate ProActive's approach to *development quality improvement* at *level 7 (C)*, due to the fulfillment of multiple quality improvement measures.

According to a product owner, the testing process was a strictly manual as there were no automated tests in place, neither in the form of integration nor unit tests (Respondent 15, ProActive, 2015). The testing process showed to be an obstacle for software development due to the postponement of testing

to the very end of the release cycle. The absence of automated tests and a continuous testing process led to rushed workflows that mainly resulted in delays or the introduction of known errors in the product. Due to the absence of automation, we assessed the *test automation* at *level 4 (B)*.

In 2015, ProActive had almost no focus on monitoring and performance-optimizing and was heavily reliant on customer feedback for feature errors and performance issues. At this time, the monitoring and incident handling processes were carried out by the operations team (Respondent 18, ProActive, 2015), who were yet to be merged with the development team. Due to the inadequate knowledge sharing between the two teams, a situation where developers had limited insights into the performance of their applications occurred. The absence of continuous performance monitoring led to an uncertainty in product performance, which hampered the team's ability to act proactively to product failures. We, therefore, rate the *incident handling* of ProActive at *level 2 (A)*, as the process of handling failures was purely reactive.

ProActive has historically followed the evolvement of Microsoft's applications. As such, ProActive used the predated version of Azure DevOps, Visual Studio Team Services (VSTS), in 2015. According to an employee, VSTS was the central configuration management system and was used to provision the internal development environments (Respondent 19, ProActive, 2015). Thus, we determine the *configuration management* of ProActive to be at *level 5 (C)*. VSTS included many of the functionalities known from Azure DevOps today, why we presume that the level of configuration management in ProActive in 2015 was already at a high level.

Despite the lack of cross-functional teams and the required technical setup, ProActive was in a favorable starting position to adopt the DevOps concept, due to the fulfillment of crucial agile principles, and their willingness to follow and adopt new upcoming trends. As such, we assess ProActive's overall previous level of maturity to be on a relatively high level compared to the amount of emphasis put on DevOps by ProActive in 2015.

6.1.2. Current Level of Maturity

6.1.2.1. Culture and Collaboration

ProActive has historically worked with agile software development, which has provided them a favorable starting position in adopting DevOps. We observe this in the embedded culture that exists

in the product team, IntraActive, today. The IntraActive team has fully adopted the ideal DevOps team structure as all employees share the same office space, and the clear distinction between Dev and Ops people is removed. According to a developer, the roles within the team are somewhat shared; however, the individual employees have separate job descriptions and focus areas that lean towards either development or operations (Respondent 1, ProActive, 2020). Nevertheless, the different tasks tend to overlap, indicating that no team member is stuck in a fixed role and, thus, the team avoids a situation where an employee possesses a large amount of tacit knowledge. Due to the merge of teams and roles, we perceive ProActive to have reached the highest level of *team organization*, thus rating them as *level 10*. *Figure 8* shows the complete assessment of ProActive's current level of maturity.

Focus Area / Level	0	1	2	3	4	5	6	7	8	9	10
Culture and Collaboration											
Communication		A				B	C			D	E
Knowledge sharing				A		B	C				D
Trust and respect							A	B	C		
Team organization		A	B						C	D	
Release alignment				A					B	C	
Product, Process and Quality											
Release heartbeat		A				B	C		D	E	F
Branch and merge			A	B		C		D			
Build automation			A	B		C					
Development quality improvement			A			B		C	D	E	
Test automation				A	B	C			D		E
Deployment automation					A	B		C			D
Release for production					A			B	C	D	
Incident handling			A					B	C	D	
Foundation											
Configuration management			A	B		C					
Architecture alignment			A					B			
Infrastructure				A			B	C	D		

Achieved level of maturity | Unachieved level of maturity | A-F | Capabilities

Figure 8: Assessment of ProActive's current level of maturity (2020)

The structure of the team has fostered a strong relationship between the employees and created a sense of a common goal within the team, where every team member has a share in the product quality. One employee expresses: *"I really see a predominant tendency, at least in our team, for everyone to have significant co-responsibility in everything we develop"* (Respondent 4, ProActive, 2020). According to a senior developer, the culture exhibits a feeling of trust and respect within the team, and the diversity of opinions and values is highly rated due to the belief that diversity facilitates meaningful discussions that improve the teamwork (Respondent 2, ProActive, 2020). Consequently, we consider ProActive to have achieved an ideal team environment, why we determine the *trust and respect* to be at the highest level possible (10).

ProActive employs multiple communication and knowledge sharing tools on both a team and organizational level. On a team level, the primary communication is through social interactions and face-to-face meetings due to the team members sharing the same office space. The team's primary communication tools are Microsoft Teams and the SharePoint platform that stores all documents created during the software development (Respondent 4, ProActive, 2020). The implementation of Microsoft Teams has changed a lot in the team culture, especially in the areas of collaboration and knowledge sharing, and one employee describes the tool as *"an essential communication tool for us"* (Respondent 4, ProActive, 2020). The team also has daily standup-meetings, weekly team meetings, and occasional retrospectives in conjunction with their agile development approach. Furthermore, the IntraActive team hosts quarterly *"Community meetings"* with their customers. At these meetings, the team highlights some of the latest features released while the customers share knowledge on how they use their intranet solution (Respondent 4, ProActive, 2020).

On an organizational level, ProActive uses Microsoft Yammer for establishing Communities of Practice to enable knowledge sharing across teams. The Yammer platform consists of numerous groups all labeled with a specific topic; for example, "Microsoft Azure" that all employees can freely join upon interest. Within the Yammer groups, new initiatives and solutions to issues are shared between the group members (Respondent 4, ProActive, 2020). As such, we assess ProActive's communication and knowledge sharing within and across teams to be on a high level of maturity due to the centralization of communication. However, ProActive has not expressed to be actively seeking to improve their communication by adopting or trying new forms of communication, and as such, we rate the *communication* of ProActive to have achieved *capability D* at *level 9*.

Internally the team is aligned concerning software releases; however, as mentioned, the IntraActive product is built on SharePoint. Despite ProActive being an official Microsoft partner, we assume that ProActive does not have extensive knowledge of the release cycle and roadmap of SharePoint. According to an employee, ProActive rely heavily on the information that Microsoft posts through their official channels for their roadmap planning and release cycle (Respondent 3, ProActive, 2020). This information is, however, not always accurate and sometimes completely lacking. This absence of information can cause situations where Microsoft develops similar applications to those of ProActive and, thus, limit the value of IntraActive, leading to wasted development time. In worst-

case scenarios, ProActive could face downtime or significant issues in their product if Microsoft releases changes that directly affect the setup of IntraActive. ProActive has measures in place to mitigate the risk of such situations occurring (Respondent 3, ProActive, 2020), but it is evident that there is a lack of external release alignment, due to external dependencies. Thus, we deem ProActive to have a high level of *release alignment* (*level 8 - B*), but as a consequence of the external dependencies, they still face challenges in this area.

6.1.2.2. Product, Process, and Quality

ProActive exhibits the use of continuous delivery as the IntraActive team can release new code daily through their Azure DevOps solution. However, the company has an agreement with their customers to only release new functionality or changes from Monday to Thursday due to the lack of support resources during the weekends, with hotfixes being an exception to this agreement (Respondent 4, ProActive, 2020).

The IntraActive team uses Git as their version control system as it is one of the two options available in Azure DevOps, with the other being TFS. Moreover, ProActive has a defined branch and merge strategy for its software development. All new code must be developed on a feature branch (Respondent 4, ProActive, 2020). A feature branch strategy is based on developers creating a specific branch when developing a feature, and then working on that feature independently from the shared or master branch. When the feature is done, it is merged into the shared or master branch, with a pull-request, to integrate the new feature into the working code base seamlessly (Shibab, Bird, & Zimmermann, 2012).

Furthermore, the team utilizes feature toggles to be able to release smaller bits of code continuously (Respondent 2, ProActive, 2020). These coding measures ensure a clean code base and should theoretically help mitigate the number of errors occurring in software development. As ProActive utilizes best practices within the area of *branch and merge* and exhibits an urge to improve in this area continuously, we rate them to have achieved the highest level possible (*10*) and thereby exceeding the defined capabilities. ProActive has not expressed the adoption of new development quality improvement measures, such as automated code quality monitoring, why we assess the level of *development quality improvement* to be unchanged at *level 7*.

The deployed Azure DevOps solution offers a myriad of tools that supports the DevOps approach from end-to-end. The platform handles everything from integration to build to deployment, and the developer solely needs to click one button to release code:

> *"DevOps is so simple in its form, and the tooling is so advanced today that you do not really have to do anything. If you were asking me five years ago, I would definitely tell you the scripts that I had to write to make things build and deploy, but today I do not have to write anything. I am just going to assist it by clicking around."* (Respondent 2, ProActive, 2020).

ProActive uses, amongst other things, *gated check-ins* as code needs to be built before deployed to an environment that prevents code from breaking running environments, thereby increasing the quality of the software development (Respondent 2, ProActive, 2020). ProActive is yet to automate their release cycle fully and, thus, achieve a continuous deployment setup, where code automatically is released to production when it surpasses the automatic testing phase. However, according to an employee, this is a conscious choice made by the team, as they wish to preserve a setup where an employee actively has to perform the release (Respondent 4, ProActive, 2020). As such, we estimate ProActive's *deployment automation* to have achieved *capability C* (*level 7*). In ProActive's continuous delivery setup, all manual processes in the release cycle are yet to be removed. The onboarding process of new customers is still a manual process that requires manual creation of required sites and resources. However, the team is actively pursuing a solution to the problem but is yet to discover the most optimal way (Respondent 4, ProActive, 2020).

ProActive makes use of automatic tests in their software development. This includes integration and unit tests that run every time new code is built as well as daily API tests that verify that all necessary endpoints are available. If an integration or unit test fails, the build is automatically abandoned, and the release is rejected (Respondent 1, ProActive, 2020). In addition to the automated tests, ProActive also conducts manual validation tests before releasing to production. These manual validation tests are a requirement set by the team and can be performed by any team member. However, ProActive does not use or expressed the desire to use recoverability or resilience tests (Respondent 1, ProActive, 2020), which means that in case of failure, the system will not selfheal or roll back to a previously functioning version. Consequently, the company still has quality measures they can improve on to

achieve a higher level of maturity within test automation, why we assess ProActive's approach to *test automation* on *level 8* (*D*).

Monitoring is primarily done through the tool, Application Insights, which is a part of the Azure toolchain. Application Insights allows the team to monitor their services' and applications' performances and help them diagnose issues. According to a senior developer, the team's use of Application Insights was limited to a more reactive approach due to the lack of logged data. As a measure to overcome, ProActive has implemented telemetric data in its applications, allowing them to get a real-time view of the product's performance (Respondent 2, ProActive, 2020). With this change, the team has shifted their incident handling from being purely reactive and reliant on customer feedback to a proactive approach allowing them to fix incidents before they get reported. The monitoring is yet to be fully automated and still requires some manual interference; hence, there are still some minor improvements to be achieved, why we rate ProActive's approach to incident handling at level *9* (*D*).

6.1.2.3. Foundation

The team structure and the merged roles that exist in ProActive ensure that the architecture of the product is aligned. According to a developer, all team members are involved in the development and release cycle (Respondent 1, ProActive, 2020). As such, we assume that there exists a continuous alignment in both the software and technical architectures. Despite the internal alignment in architecture, there still exist challenges in terms of external alignment due to the dependency on SharePoint. Changes in the architecture of SharePoint might not align with the architecture of IntraActive due to a lack of information regarding new releases from Microsoft. As ProActive has internal alignment but still is hampered by significant external technical dependencies that influence their product, we evaluate the level of *architecture alignment* at *level 7* (*B*).

For configuration management, the company utilizes an automated approach through Azure DevOps as well as other tools within the Azure Portal. Configuration management streamlines the delivery of software and applications by automating the build-out of systems quickly and efficiently. A few years ago, configuration management was something organizations actively had to devise a strategy and implementation plan for. Nowadays, there is an extensive collection of tools available that handle configuration management seamlessly and automatically, specifically for Platform-as-a-Service

(PaaS) solutions (Whyte, Stasis, & Lindkvist, 2016). Every aspect of the IntraActive product is stored and managed inside the Azure Portal. This includes databases, middleware, SDKs, APIs, and more (Respondent 2, ProActive, 2020). Thus, the use of the Azure Portal secures a strong relationship between the required configuration items, while simultaneously supporting the items with version control, as this is a feature within the toolchain of Azure. As such, the tools provide ProActive with a relatively high level of maturity concerning configuration management, contrary to the low priority put on the discipline. However, the same required capabilities to reach a high level of configuration management were already available in the predecessor to Azure DevOps, VSTS; hence we rate ProActive's *configurations management* to be unchanged at *level 5*.

ProActive operates with multiple deployment environments: development, test, first-release, and production (Respondent 4, ProActive, 2020). The development environment is, as the name suggests, restricted to development. The test environment is where the validation tests are performed, and the first-release environment is ProActive's internal intranet solution hence a production-like environment. The many deployment environments streamline the development cycle and seek to help minimize the number of errors. However, each of ProActive's customers has its own production environment that may contain many different customizations, which could impact new releases and, in the worst case, make them fail. Some errors can only be detected in the final production environment despite the many code quality initiatives (Respondent 4, ProActive, 2020).

ProActive's infrastructure allows them to deploy new applications directly to each customer tenant quickly. As SharePoint is a preconfigured platform, ProActive is not required to configure or provision customer environments, which is another benefit of the infrastructure. Furthermore, some of the components inside SharePoint can be reused or evolved by ProActive, thus, saving them development time. Contrariwise, specific decisions on application architectures will be predefined by SharePoint due to the compatibility issue. According to an employee, the on-going requirement to maintain compatibility with SharePoint can conflict with existing parts of the product, which means that the team sometimes has to restructure existing code for it to function on the SharePoint platform (Respondent 3, ProActive, 2020). Besides occasional compatibility issues, the infrastructure of IntraActive allows for ProActive to deploy new applications fast and without friction, while exploiting the many benefits of SharePoint and Azure. Therefore, we rate ProActive's *infrastructure* at *level 8* (*D*).

6.1.3. Summary

ProActive has exploited its favorable starting position for adopting DevOps and exhibits a mature and robust adoption of the DevOps concept today. They have since 2015 merged the two traditional teams into one cross-functional team where the roles and tasks overlap. Furthermore, they have defined clear communication and knowledge-sharing protocols through tools such as Microsoft Teams and Yammer, which have helped facilitate the development of the advantageous DevOps culture. *Figure 9* shows a detailed view of maturity development in ProActive.

Focus Area / Level	0	1	2	3	4	5	6	7	8	9	10
Culture and Collaboration											
Communication		A				B	C			D	E
Knowledge sharing				A		B	C				D
Trust and respect							A	B	C		
Team organization		A	B						C	D	
Release alignment				A					B	C	
Product, Process and Quality											
Release heartbeat		A				B	C		D	E	F
Branch and merge			A	B		C		D			
Build automation			A	B		C					
Development quality improvement			A			B		C	D	E	
Test automation				A	B	C			D		E
Deployment automation					A	B		C			D
Release for production					A			B	C	D	
Incident handling			A					B	C	D	
Foundation											
Configuration management			A	B		C					
Architecture alignment			A					B			
Infrastructure				A			B	C	D		

Previous level of maturity | Current level of maturity | A-F Capabilities

Figure 9: ProActive's DevOps maturity progression

On the technical side of DevOps, ProActive has moved from a fixed release cycle to a continuous delivery setup. Their connection to Microsoft and their willingness to follow the new initiatives and evolvements within the applications from Microsoft have improved ProActive's approach to development. The Azure Portal today is a big part of the development and operation in the IntraActive product as it provides the tools the necessary tools for continuous development and improvement. Especially monitoring and optimization have seen significant improvements in ProActive as they have shifted their reactive incident handling to an analytical and proactive approach that allows them to fix errors before they get reported by customers. Even though the company has positively progressed in terms of DevOps, there are still multiple areas they can improve on, such as configuration management and test automation.

6.2. Topdanmark Maturity Analysis

In this section, the maturity level of Topdanmark will be assessed for two states in time: a previous state (2015) and the current state (2020). The assessment of the previous state will serve as a reference point, and thus the assessment of this state of maturity will be more narrow. The assessment of the current state will be more in-depth to provide an insight into the current development cycle, culture, and technical infrastructure of the company. All focus areas of the Focus Area Model are included in the maturity assessment, but will not all be explicitly described, due to the large number of focus areas included in the model.

6.2.1. Previous Level of Maturity

In 2015, Topdanmark's organizational structure and team organization were primarily affected by the shift towards an agile organization three years prior. The IT-department was split into smaller teams that each focus on their area, which was something that the employees still were getting used to (Respondent 20, Topdanmark, 2015). According to an employee, the organization was split into distinct silos, with a clear separation between the development and operations teams:

> *"There is and has always been a sort of 'us' and 'them' thing between development and operations. In operations, you can hear them say 'them up on the first floor,' maybe it is the business, but it is also development, so there definitely is some 'us and them.' We also say 'them down in operations', so down in operations (...) So the way we refer to each other in is not promoting shared core values. It (the culture) is maintaining this division"* (Respondent 20, Topdanmark, 2015).

Furthermore, it is apparent that Topdanmark did not have any cross-functional teams that included developers and operations team members (Respondent 21, Topdanmark, 2015; Respondent 22, Topdanmark, 2015). However, as Topdanmark had a tester assigned to every development team, the *team organization* is rated at *level 2* (*B*). *Figure 10* shows the complete assessment of Topdanmark's previous level of maturity.

Focus Area / Level	0	1	2	3	4	5	6	7	8	9	10
Culture and Collaboration											
Communication		A				B	C			D	E
Knowledge sharing				A		B	C				D
Trust and respect							A	B	C		
Team organization		A	B						C	D	
Release alignment				A					B	C	
Product, Process and Quality											
Release heartbeat		A				B	C		D	E	F
Branch and merge			A	B		C		D			
Build automation			A	B		C					
Development quality improvement			A			B		C	D	E	
Test automation				A	B	C			D		E
Deployment automation					A	B		C			D
Release for production					A			B	C	D	
Incident handling			A					B	C	D	
Foundation											
Configuration management			A	B		C					
Architecture alignment			A					B			
Infrastructure				A			B	C	D		

Achieved level of maturity | Unachieved level of maturity | A-F Capabilities

Figure 10: Assessment of Topdanmark's previous level of maturity (2015)

According to an employee, due to the average tenure of employees being more than ten years, the culture was very fixed, and there was a general idea of not being able to change the culture to support the cross-functional teams without putting in a great effort. The *communication* between the developers and operations was almost non-existent, with the majority of the communication being finger-pointing in the light of incidents, suggesting an achieved capability *C* at *level 6* (Respondent 22, Topdanmark, 2015). Topdanmark had implemented a cross-organizational knowledge sharing platform, IBM's Connections (now HCL Connections), but according to several of the employees in Topdanmark, there were not many employees that used it, and there did not exist a sufficient overview of what information was available on the platform (Respondent 20, Topdanmark, 2015; Respondent 22, Topdanmark, 2015). At the time, there was no communication platform specific to the organization, so it was up to the individual team to choose how they preferred to communicate. The *knowledge sharing* of Topdanmark is, therefore, evaluated to be at *level 3* (*A*).

The release heartbeat of Topdanmark in 2015 was fixed across the organization. Since Topdanmark was following the Top-Up process, they had a total of 10 releases per year with one at the end of every month (excluding December and July). As such, it is clear that Topdanmark did not have any continuous delivery process with ongoing deployments to production at that moment, but the *release heartbeat* was fixed nonetheless (*level 5 - B*). A lot of the processes in Topdanmark were done manually at the time, including testing (Respondent 20, Topdanmark, 2015; Respondent 22,

Topdanmark, 2015; Respondent 23, Topdanmark, 2015). However, the ambition of transitioning towards a continuous delivery model with automated processes was clear: *"Now we want to shift towards using continuous delivery, and then the test process will become automated and ongoing during the entire process"* (Respondent 24, Topdanmark, 2015).

Some processes were automated in Topdanmark at the time: *build* and *configuration management* were automated through the tools used in Topdanmark and are, therefore, assessed to have achieved *capability B* (*level 3*). Furthermore, the branching and merging were done using Git. We deem that Topdanmark utilized a feature branch strategy as a branching/merging strategy (Respondent 24, Topdanmark, 2015), and we assign the *branch and merge* focus area to *level 5* (*C*). This branching and merging strategy suggests that Topdanmark had gated check-ins on code, thereby fulfilling *capability C* (*level 7*) of the *development quality improvement* focus area.

Subsequently to features being deployed to the production environment, the *incident handling* of Topdanmark was entirely reactive (*level 2 - A*). Accordingly, all incidents that occurred from the production environment was either reported by the system, as a result of malfunctioning code or system errors, or by customers that contacted the support team in Topdanmark. A developer described the handover process between development and operations as: *"The code goes into production, and when something goes wrong, then you meet operations"* (Respondent 24, Topdanmark, 2015). As such, the communication between the two teams was limited, and the incident handling was often precarious due to the lack of knowledge sharing (Respondent 21, Topdanmark, 2015).

The technical infrastructure of Topdanmark was largely affected by the mainframe, as almost all systems were dependent on the mainframe to some extent (Respondent 20, Topdanmark, 2015). While the mainframe was considered a problem and a substantial obstacle in the pursuit of continuous delivery, it is accentuated that: *"all applications that use backend services are not isolated, we are not going to find ourselves in a situation where we exhibit services that do not have backend services"* (Respondent 25, Topdanmark, 2015). Due to all services being dependent on the mainframe, Topdanmark had an alignment between the software and technical architecture since any software with backend calls that was not supported by the mainframe would not function. As such, the *"old mainframe"*, as it is referred to (Respondent 20, Topdanmark, 2015), was a significant obstacle for Topdanmark in implementing continuous processes, but it ensured *architecture alignment* between

the software and technical layers (*level 2 - A*). The *infrastructure* focus area is mainly concerned with the technical environments of the organization, and as we deem Topdanmark to have a *partly automatically provisioned infrastructure* they are assigned a *B* (*level 6*) for this capability.

6.2.2. Current Level of Maturity

6.2.2.1. Culture and Collaboration

Topdanmark has not successfully adopted a DevOps culture as it is described in the available literature, where the silos between the development and operations team are broken down to create a united team that covers the entire process from development to monitoring the application in production. They have, however, adopted the DevOps concept on their own terms to fit their specific situation. The reason behind Topdanmark not having adopted the advocated culture could be explained by the organization's large size, a very high average tenure of employees, lack of skills, management decisions, or something completely different, which will be further investigated later in the analysis.

The communication within the development teams in Topdanmark is defined as structured according to the maturity model of Feijter et al. (2018), due to the teams following Scrum and having daily stand-ups, retrospectives after release to production, and including product managers in the entire process of development. However, the communication between the development and operations teams is kept on the minimum, and there is a rigid handover process when a finished feature or function is developed and is going to production. Due to the lack of operations teams inclusion in the daily stand-ups and retrospectives, the *communication* in Topdanmark is not assessed to fulfill all requirements needed for *capability D*. We, Therefore, determine the *communication* at *level 8*. *Figure 11* shows the complete assessment of Topdanmark's current level of maturity.

As described in the literature, in any organizational DevOps setup, knowledge sharing between entities is crucial. In the case of Topdanmak, it is even more accurate due to the handover process between development and operations. As the operations teams are not included in any of the development processes, and thereby necessarily do not have any knowledge concerning the developed applications, the knowledge sharing in the handover process is crucial. When asked about the knowledge sharing process in Topdanmark, a manager replies:

" That is something we are not so good at in general in Topdanmark. We are extremely helpful and help each other whenever one is in need (...), but we are not so good at systemizing knowledge and informing other people about what is going on so they can learn from it." (Respondent 10, Topdanmark, 2020).

Focus Area / Level	0	1	2	3	4	5	6	7	8	9	10
Culture and Collaboration											
Communication		A				B	C			D	E
Knowledge sharing				A		B	C				D
Trust and respect							A	B	C		
Team organization		A	B						C	D	
Release alignment				A					B	C	
Product, Process and Quality											
Release heartbeat		A				B	C		D	E	F
Branch and merge			A	B		C		D			
Build automation			A	B		C					
Development quality improvement			A			B		C	D	E	
Test automation				A	B	C			D		E
Deployment automation					A	B		C			D
Release for production					A			B	C	D	
Incident handling			A					B	C	D	
Foundation											
Configuration management			A	B		C					
Architecture alignment			A					B			
Infrastructure				A			B	C	D		

Achieved level of maturity | Unachieved level of maturity | A-F Capabilities

Figure 11: Assessment of Topdanmark's current level of maturity (2020)

Topdanmark is determined to have a constructive culture, but they lack some engagement and structure in their knowledge sharing process. However, knowledge sharing has been improved in the past year. In December 2018, Topdanmark introduced Office 365, and with it, Microsoft Teams into the organization, which allows for a centralized platform for knowledge sharing. The introduction of a shared platform has taken the knowledge sharing between the teams to *"another level"* (Respondent 10, Topdanmark, 2020). Topdanmark has improved the *knowledge sharing* within the organization to *level 5* by achieving *centralized knowledge sharing* (*B*), but still has much room for improvement. To mature their knowledge sharing processes, active steps can be taken towards a more *active knowledge sharing* culture, where development and operations employees engage in active discussions about the possibilities of their solutions or provide training to each other. Actively engaging such knowledge sharing processes will help the teams to fully grasp the work processes of each other, which likely will result in better teamwork and a more efficient handover process.

As a step towards a higher level of knowledge sharing, some employees of Topdanmark have established communities of practice, where common passions between the employees are shared. Communities of practice are an excellent way of sharing knowledge and are something Feijter et al. (2018) include as the highest capability of knowledge sharing in the Focus Area Model. According to an employee, however, some of these communities of practice have particular prerequisites, such as attendees knowing specific coding languages (Respondent 10, Topdanmark, 2020). Starting these communities of practice is an initiative towards a more thriving DevOps culture, but having prerequisites of existing knowledge for attendees could profoundly influence the type of employee that attends. With the prerequisite of knowing coding languages, there is a high chance that the attendee base will consist primarily of developers. If Topdanmark has aspirations of creating DevOps communities of practice to enhance the knowledge sharing between development and operations teams, the discussed areas of interest should perhaps be of a more general characteristic so they can harvest the advantage of having diverse employees from different teams participate. While the presence of communities of practice suggests that Topdanmark has achieved *capability D* of *knowledge sharing*, they cannot advance to this level without fulfilling the requirements of *capability C*, where more steps towards an *active knowledge sharing* culture have to be taken.

Another element that is presented in the literature and included as a focus area in the Focus Area Model, as an essential part of a DevOps culture, is trust and respect amongst employees. Topdanmark has a decentralized organizational structure, where responsibility is allocated to the individual teams and units. There is an ingrained trust from management in the ability of each team to make the decisions that are most appropriate for them. From the interviews, it is apparent that there is a high level of respect between the employees, leading to an assessment of *level 7* (*B*) in *trust and respect*. The high level of respect could be connected to the high tenure of employees in Topdanmark. Following the data presented in the interviews, the average tenure of employees in the IT-department of Topdanmark is around 10-20 years (Respondent 12, Topdanmark, 2020). With an average tenure of more than a decade, employees have been working together very long and have, all things held constant, done their job well enough to be respected for it.

Feijter et al. (2018) introduce core values and a shared goal as a driver of trust and respect. The development and operations teams should be rewarded as a group when a release is successful, as well as having a high level of transparency and prevent blaming in faulty situations. As of now, it is

clear that there is no clear vision of a shared goal between the teams and that Topdanmark has not implemented any group rewards when a feature or application finishes the cycle from development to functioning in production. The development and operations teams have individual measures and rewards for the accomplishments, varying across teams, and management has not focused upon creating a shared goal (Respondent 6, Topdanmark, 2020).

Because of the software development process being divided into the Top-Up process and the CD-process, the internal release alignment is difficult for Topdanmark to achieve. This is mainly due to the two processes having adverse deployment patterns and teams not adhering to a common sprint cadence (Respondent 11, Topdanmark, 2020). However, Topdanmark is deemed to have improved on the alignment within each of the processes since 2015. We therefore assess Topdanmark's *release alignment* to *level 6* with an inability to achieve *capability B*, currently, due to the diverse processes.

6.2.2.2. Product, Process, and Quality

Following Topdanmark's decentralized organizational structure, the process of developing and following code through to production and monitoring differs from team to team. However, there are similarities across the teams. In the development process, Git is utilized by the vast majority of developers across the organization for version control and merging code. According to a manager, most teams have implemented a feature branch strategy for developing code to integrate new features into the working code base seamlessly (Respondent 10, Topdanmark, 2020). The teams in Topdanmark are very aware of shortening the lifespan of the created feature branches to a maximum lifespan of 2-3 days. The advantage of having short-lived branches is due to the minimized risk of incurring conflicts and bugs when integrating with the master branch (Shibab et al., 2012).

When code is committed to the master branch in Topdanmark, it does not go directly to the production environment but instead to a staging environment from which deployment can take place. Topdanmark further uses Jenkins for integrating the code into production. Due to the possibilities a tool like Jenkins provides, Topdanmark could potentially remove the staging environment from their process and push code directly to production, but they prefer to complete the step of pushing to production manually. In the process of integrating code, Topdanmark uses Docker and Jenkins to build the software automatically. We, therefore, rate *build automation* at *level 5* (*C*). However, according to the DevOps specialist in IT operations, not all teams' developing processes support the

automatic build, and some do not have the competences or knowledge to include it in the process (Respondent 12, Topdanmark, 2020).

When releasing developed code to production, feature toggles are often used in a DevOps setup. Feature toggles provide the functionality of hiding published functionality from users of the system. Released features are then hidden behind a wall, but the code is pushed to production and working correctly. By using feature toggles, small parts of a feature can be released to production, without being visible to users, making it possible to continuously test and monitor parts of the solution while the rest is being built (Respondent 13, Topdanmark, 2020). According to an employee, the functionality of feature toggles is available in the organization, and some teams already use it, but it is constrained, and it is not something that has been introduced widely to the organization (Respondent 13, Topdanmark, 2020). However, as feature toggles are a possibility in the development of Topdanmark and the earlier capabilities are fulfilled by the feature branch strategy, we have assessed the *branch and merge* focus area to have achieved *capability D* (*level 7*).

Following the completed development of a feature, it is tested. The testing process in Topdanmark varies from team to team and is not perceived to be very structured. Most of the teams are in a stage between manual and automated testing, utilizing both. Most teams have a dedicated tester or a product owner that tests developed functionality to make sure it meets the requirements. Some teams have thorough integration tests, and some use Test Driven Development (TDD) in the development process. There are instances of unit tests, both manual and automatic. Non-functional tests, such as baseline-tests, have also been introduced in some teams but left out in others. As such, the testing capabilities and processes vary a lot between the teams, and the perception is that no teams know entirely what the possibilities are within the organization regarding testing and that the teams are not aware of the processes of other teams. A manager mentions that *"either the teams have an interest in testing, or else they have no interest in test and have not built anything yet"* (Respondent 10, Topdanmark, 2020). While the testing within Topdanmark varies between teams, they are determined to fulfill the requirement of *automated systematic testing*, and we accordingly rate the *test automation* at *level 5* (*C*).

To centralize, structuralize, and to get an insight into the quality of the developed code within Topdanmark, they have recently implemented SonarQube (Respondent 10, Topdanmark, 2020).

SonarQube is an open-source tool that can perform continuous code inspection and quality monitoring. The implementation of SonarQube has made it possible for Topdanmark to analyze all the code in their Git repositories to identify if any significant shortcomings or low quality is apparent in the existing code. Aside from being used as an analytic tool on existing code, SonarQube is also utilized when developing new code to ensure high quality in any releases. As Topdanmark both have *gated check-ins* and *automated code quality monitoring*, the *development quality improvement* is rated to have achieved *capability D* (*level 8*). Topdanmark recently used SonarQube to ascertain how many of their projects that had a code coverage of 80% or higher in unit tests, which exemplifies one of the possible uses of the tool (Respondent 10, Topdanmark, 2020).

When a feature or application is developed and tested, the next step in the process is releasing and deploying the application to the production environment. Primarily due to technical dependencies and culture differences, the release alignment and deployment phase of Topdanmark is characterized by complexity. Firstly, the IT architecture of Topdanmark is built around a legacy system but has, during the later years, been slowly restructured towards supporting a newer technological model. The backend is split between their mainframe, hosted locally in Topdanmark, and AWS that allows for a cloud-based and serverless backend. The decision of which backend is utilized relies heavily on the application and dependent systems. Secondly, according to an employee, the culture is still imprinted by a *"us and them"*-view between the development and operations team (Respondent 10, Topdanmark, 2020).

Further, the traditional release culture of releasing code in large batches has yet to transition to a more continuous release culture, where features and applications are sliced, for a large part of the developing teams (Respondent 11, Topdanmark, 2020). These two factors of technical dependencies and culture arguably have an impact on the release and deployment process for Topdanmark. However, as Topdanmark has *continuous delivery* on the CD-process, the *deployment automation* is assessed to be at *level 7* (*C*). Currently, Topdanmark cannot succeed beyond *level 7* as the organization has to adhere to the regulatory compliance of segregation of duties that hinders them from implementing continuous deployment.

The release heartbeat of Topdanmark also differs depending on whether the release takes place as part of the Top-Up process or CD-process. On the CD-process, where the architecture and tools allow

for it, the ICE-WEB team can release as often as they determine necessary. In the data, there is a slight inconsistency in how often the team can deliver features and hotfixes. We estimate it to be anywhere between several times a day to once a week, which, regardless, is a lot more often than in the past. In contrast, releases on the Top-Up process are dependent on the mainframe, and the release heartbeat in unchanged since 2015 with deliveries about once a month (10-12 times a year). According to one employee, frontend features on the Top-Up process would be able to deliver more often than once a month, but since the frontend solution, and perhaps any new or changed APIs, are dependent on the backend, it would not make sense to release the frontend (Respondent 8, Topdanmark, 2020). Thus, frontend solutions on the Top-Up process are determined to be deployed once a month because of the dependencies.

Topdanmark is focusing on improving its abilities to slice functionality for releases. As mentioned, the organization is still affected by the traditional mindset of batch releases. Slicing frontend functionality is less critical on the Top-Up process than the CD-process, due to the backend batch release model. However, on the CD-process, the ability to slice functionality and release it sequentially has shown to be advantageous for Topdanmark. According to a scrum master, some teams within the CD-process are good at slicing functionality and release to production continuously using feature toggles, but some teams still see the new way of slicing functionality as a challenge (Respondent 11, Topdanmark, 2020).

Nonetheless, we assess the *release heartbeat* of Topdanmark at *level 7*. The reason behind placing Topdanmark between two capabilities is due to the added focus on gradual releases in the near future. An employee informs us that Topdanmark is looking to implement a tool to handle feature toggles that allows them to release to specific customers at a time, and thereby introduce gradual releases (Respondent 13, Topdanmark, 2020). Thus, they have not achieved the gradual release capability yet, but are arguably very close to achieving it.

The different environments of Topdanmark are monitored by their support team, Hawks, who are a vital part of operations. When incidents occur in any of the environments, Hawks determine where the incident originated and contact the responsible team or fix it themselves if possible. As of now, Hawks are trying different tools and models to be more proactive on incidents through the monitoring of environments and systems. However, according to a developer in Hawks, the majority of incidents

are reported by customers using the applications (Respondent 9, Topdanmark, 2020). As such, the majority of the incident handling in Topdanmark is reactive, but there is a focus on transitioning to become more proactive. Therefore, we do not assess Topdanmark to have achieved the capability of *proactive incident handling*, as the majority is still reactive, and we accordingly place them at *level 6*.

6.2.2.3. Foundation

Due to the existence of both the Top-Up process and the CD-process, it is complicated to determine the architecture alignment within Topdanmark. The alignment between the technical structure of the CD-process and the application and business layer is adequate, less so for the Top-Up process. The longstanding mainframe and database have shown to be a ball and chain for Topdanmark and a challenge for architecture alignment. Currently, they are trying to develop frontend in a new and modern way but based on archaic thinking of backend. The software and technical architecture alignment are existing, but the overall maturity of their architecture alignment is specifically low. Almost all the interviewed employees are aware of the challenges the mainframe poses and see the motivation in moving to a more compatible system. As more people in Topdanmark are becoming aware of the problems, and the architecture alignment gains an increased focus, we have advanced the *architecture alignment* by one level to *level 3*. Unchanged from the previous maturity assessment, the architecture alignment is still the weakest focus area of all and should, therefore, act as a red light for Topdanmark as something that needs increased attention.

As a central part of the technical foundation in a DevOps setup, lies configuration management. It was not possible to obtain any data in the interviews regarding the configuration management within Topdanmark directly. However, through the use of different tools, configuration management should be automated. Within Topdanmark, Docker is responsible for a large part of the configuration, and Git is responsible for version control. Specifically, for the CD-process, the automated configuration management tool, AWS Config, is available through the use of AWS. As such, we determine Topdanmark to have achieved *level 5* (*C*) of *configuration management*. Notably, configuration management is not something Topdanmark has expressed to focus on, but rather something that is simply handled through the implemented tools.

The provisioning and infrastructure of the environments within Topdanmark is, once more, challenging to determine, due to the separated processes. In the CD-process, the infrastructure is automatically provisioned and administered through the serverless setup provided by AWS. It is important to emphasize that this does not correctly depict the entire organization, as the Top-Up process is tied to the mainframe, where more manual provisioning of the infrastructure is needed. However, the maturity level of the infrastructure of environments within Topdanmark is determined by them supporting automatic provisioning through the tools that are available within the organization, and therefore accordingly assessed to *level 8* (*D*).

6.2.3. Summary

Topdanmark has managed to adopt certain aspects of the DevOps concept in their processes to suit their specific situation. The culture and organizational structure are currently not adhering to the principles that are presented in the available literature, where silos are broken down, and responsibility of developed features is shared from development to production and monitoring. In the five years from 2015 to 2020, Topdanmark has not drastically developed in the area of culture and collaboration, from the parameters in the Focus Area Model. *Figure 12* shows a detailed view of the maturity development in Topdanmark.

Focus Area / Level	0	1	2	3	4	5	6	7	8	9	10
Culture and Collaboration											
Communication		A				B	C			D	E
Knowledge sharing				A		B	C				D
Trust and respect							A	B	C		
Team organization		A	B						C	D	
Release alignment				A					B	C	
Product, Process and Quality											
Release heartbeat		A				B	C		D	E	F
Branch and merge			A	B		C		D			
Build automation			A	B		C					
Development quality improvement			A			B		C	D	E	
Test automation				A	B	C			D		E
Deployment automation					A	B		C			D
Release for production					A			B	C	D	
Incident handling			A					B	C	D	
Foundation											
Configuration management			A	B		C					
Architecture alignment			A					B			
Infrastructure				A			B	C	D		

Previous level of maturity | Current level of maturity | A-F Capabilities

Figure 12: Topdanmark's DevOps maturity progression

However, the implementation of Microsoft Teams as a cross-organizational communication platform has improved their knowledge sharing. The implementation of a central communication channel has also indirectly improved their team organization since it has made it easier for employees to get into contact with employees from different teams. In 2016 the DevOps team within Topdanmark was created. Following the decision of Topdanmark's management on not merging development and operations, the creation of the DevOps team was an essential step towards a DevOps culture. The reason for Topdanmark not having successfully adapted a DevOps culture can be tied to several factors, which we will further discuss as a part of the thesis.

Since the introduction of DevOps by Topdanmark's management in 2015, the organization has significantly advanced within the process, product, and quality section of the Focus Area Model. The improvement of processes within developing, testing, deploying, and monitoring is primarily tied to the implementation of several useful tools in the organization. Tools such as Jenkins, Docker, and SonarQube have significantly increased the efficiency and quality of processes tied to developing and delivering the software of Topdanmark. Furthermore, the introduction of AWS and serverless development have made it possible for the organization to continuously deliver software and automatically provision the infrastructure in their ICE-WEB team.

Furthermore, the incident handling of Topdanmark has become partly proactive with the introduction of real-time monitoring. Topdanmark's mainframe and Top-Up Process is still a significant obstacle in the process of progressing in DevOps maturity. The management and employees of Topdanmark understand the complications tied to having the mainframe, and they are steadily trying to phase it out. However, the management acknowledges that it will be many years before the mainframe is completely phased out.

While Topdanmark, during the previous five years, has positively progressed in terms of DevOps maturity, there are still multiple areas they can improve on. Specifically, the culture, organizational structure, mindset, and technical architecture are obstacles in progressing and becoming more mature within DevOps.

6.3. Visual Maps

In the visual maps, events that are connected to the DevOps process, directly or indirectly, in the two case organizations are displayed as boxes. Events that have occurred in the case companies are placed on the map according to the data, and to the best of our knowledge, but we cannot ensure that the context of all events, particularly the timeframe of said events, are entirely accurate. During the retrospective interviews, each case organization representative was presented with an event timeline and had the opportunity to comment on the timeline to mitigate the risk of inaccuracy in the visual process maps. The exact context of the parameters in the visual maps is explained in detail in the following. *Figure 13* and *Figure 14* show the visual maps of ProActive and Topdanmark, respectively.

6.3.1. Issue Domains

Most process theories relate the events to different categories regarding the outcome (Winkler & Günther, 2012). Langley (1999) refer to these categories of domains in which events occur as issue domains. As our analysis is focused upon investigating the processes within DevOps, we have chosen three relevant issue domains: Organization, IT, and Business.

In the visual maps, the issue domain of organization encapsulates events that relate to organizational structure, culture, communication, collaboration, and knowledge sharing. The organizational issue domain closely resembles the "culture and collaboration" area of the Focus Area Model. The IT issue domain encompasses all events relating to the technical infrastructure, and the implementation and utilization of tools. Lastly, the business domain contains all events that are related to business, management decisions, or external assistance (e.g., hiring of external consultants).

Due to many aspects of DevOps impacting multiple of the issue domains, the event will be placed in the issue domain where it is considered most dominant. However, events can be placed in more than one issue domain if determined necessary to become hybrid mappings. An example of a hybrid mapping is in the event of implementing Office 365 in the organizations. In this example, it introduces a different way of communicating internally in the organization (organization); it is an extensive collection of tools that needs to be implemented and assisted by the existing technical infrastructure (IT); and it could be an effort to centralize communication, increase efficiency, and decrease costs of

inter-organizational communication (business). In the visual maps, we illustrate hybrid mappings using swim lanes and overlapping boxes.

6.3.2. Boxes

All the events or states are illustrated graphically as boxes. The form of the boxes indicates which type of event it represents. To avoid introducing unnecessary complexity into the visual maps, there are only four types of events: Decisions or initiatives, activities, external dependencies, and general information or state.

A decision or initiative regarding DevOps within the organization is represented with a round-cornered rectangle. Similarly, an activity is represented by a sharp-cornered rectangle. External dependencies, such as a third-party supplier of software or regulatory requirements, are depicted as ovals. Lastly, hexagons are used to declare any general information or state about the company that is relevant for the DevOps processes within. The majority of the boxes are illustrated in a fixed size. However, the boxes can differ in width, to illustrate the timeframe of the event, or in length, to illustrate hybrid-mapping. In the further analysis, we reference events and states in-text with the use of *italics.*

6.3.3. Direct and Indirect Relationships

The events in the visual map all have a connection to another event, be that precedent or subsequent. These relationships between events are illustrated as one of two lines. We differentiate between direct (solid line) and indirect (dashed line) relationships. *Direct relationships* exhibit a precise temporal sequence and causal dependency and can, therefore, also be regarded as transitions that form the process (Winkler & Günther, 2012). An example of a direct relationship is the decision to introduce the DevOps concept in an organization (event A) leads to a merging of the development and operations teams (event B). As such, event A is the predecessor of event B, and event B would not have occurred if event A had not occurred beforehand.

An *indirect relationship* can be regarded as a weaker causal dependency, where causality is used in a probabilistic way (Winkler & Günther, 2012). For example, the decision to merge the development and operations teams (event B) might increase the likelihood of a developer or operations employee

with high tenure to leave the company (event C). In this scenario, event B increased the probability of event C to happen, but event C could have occurred regardless of event B having occurred.

6.3.4. Maturity Indicators

To intertwine process theory with maturity theory and to further the understanding of which processes influence the maturity level of an organization, maturity indicators are included in the visual maps. The maturity indicators are adjoined to certain events that are determined to have a notable impact on the DevOps maturity of the organizations. The type and severity of impact on maturity are based on both the data obtained from the organizations and the capabilities defined by Feijter et al. (2018). The visual maps include three types of maturity indicators: Inhibiting or interrupting elements, facilitating elements, and augmenting elements.

Inhibiting or interrupting elements are internal or external events that inhibit interest in, slow down, or interrupt ongoing DevOps processes. The inhibiting or interrupting elements are depicted as negatively signed (-) symbols on events. An example of such an event is the *large turnover of employees* in ProActive in 2016 that resulted in lost knowledge and skills and the restructuring of a completely new team.

Facilitating elements are internal or external events that facilitate or accelerate DevOps maturity of the organizations. Facilitating elements are shown on our visual maps using positive-signed (+) symbols on events. An example of an event with a facilitating element is the implementation of SonarQube in Topdanmark. While implementing SonarQube is not something that revolutionizes the maturity of Topdanmark, the introduction of continuous code inspection will increase the code quality and is a step towards a more DevOps mature organization.

Lastly, augmenting elements are internal or external events that have a close positive correlation with the DevOps maturity of the organizations. Augmenting elements are depicted as double-positive-signed (++) symbols on events. An example of such is the complete implementation of continuous delivery in ProActive. The implementation of continuous delivery is something that has dramatically impacted the DevOps processes of the organization positively, both according to the employees of ProActive and the maturity model of Feijter et al. (2018).

6.4. Visual Map of ProActive

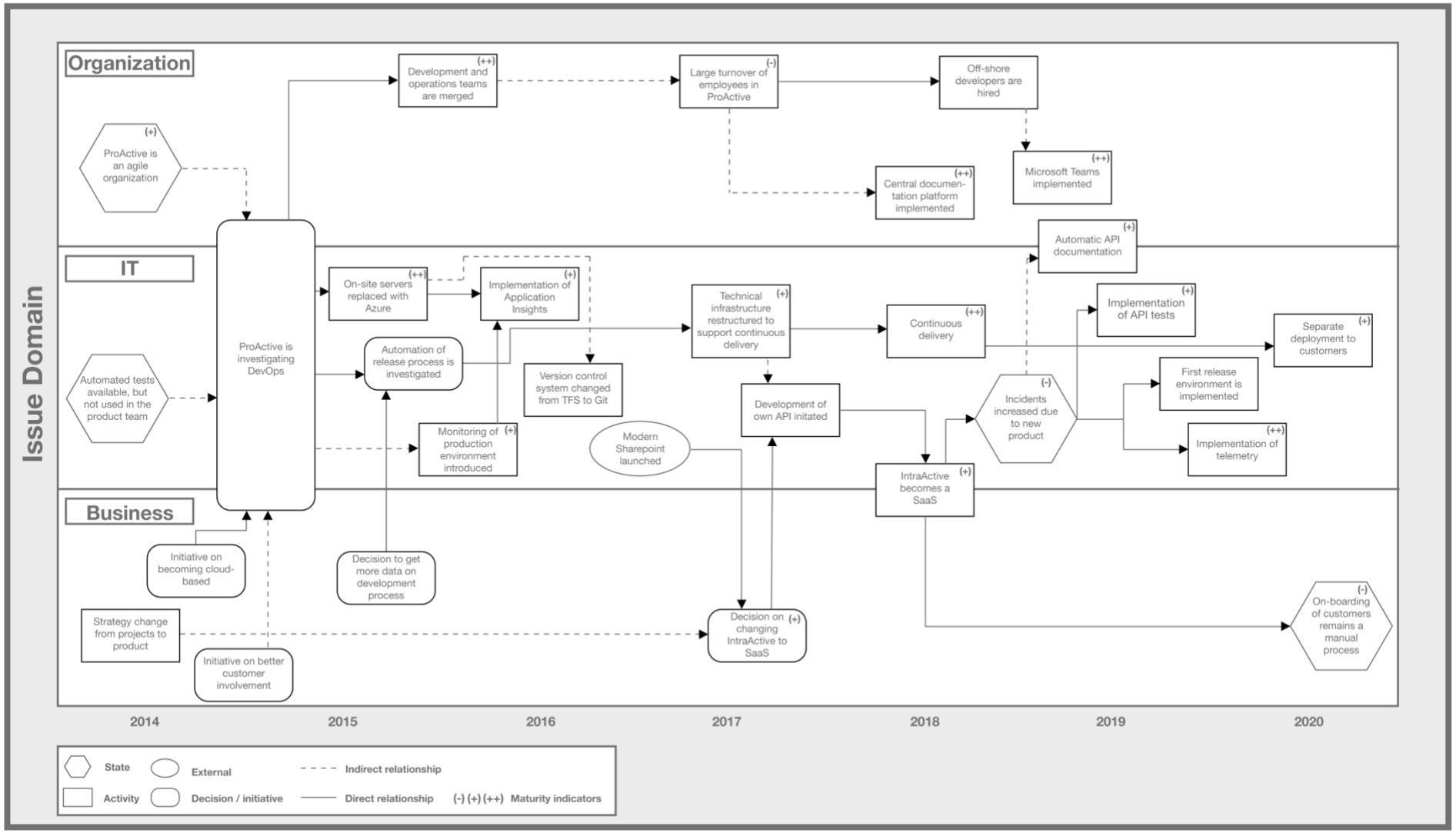


Figure 13: Visual map of ProActive

6.5. Visual Map of Topdanmark

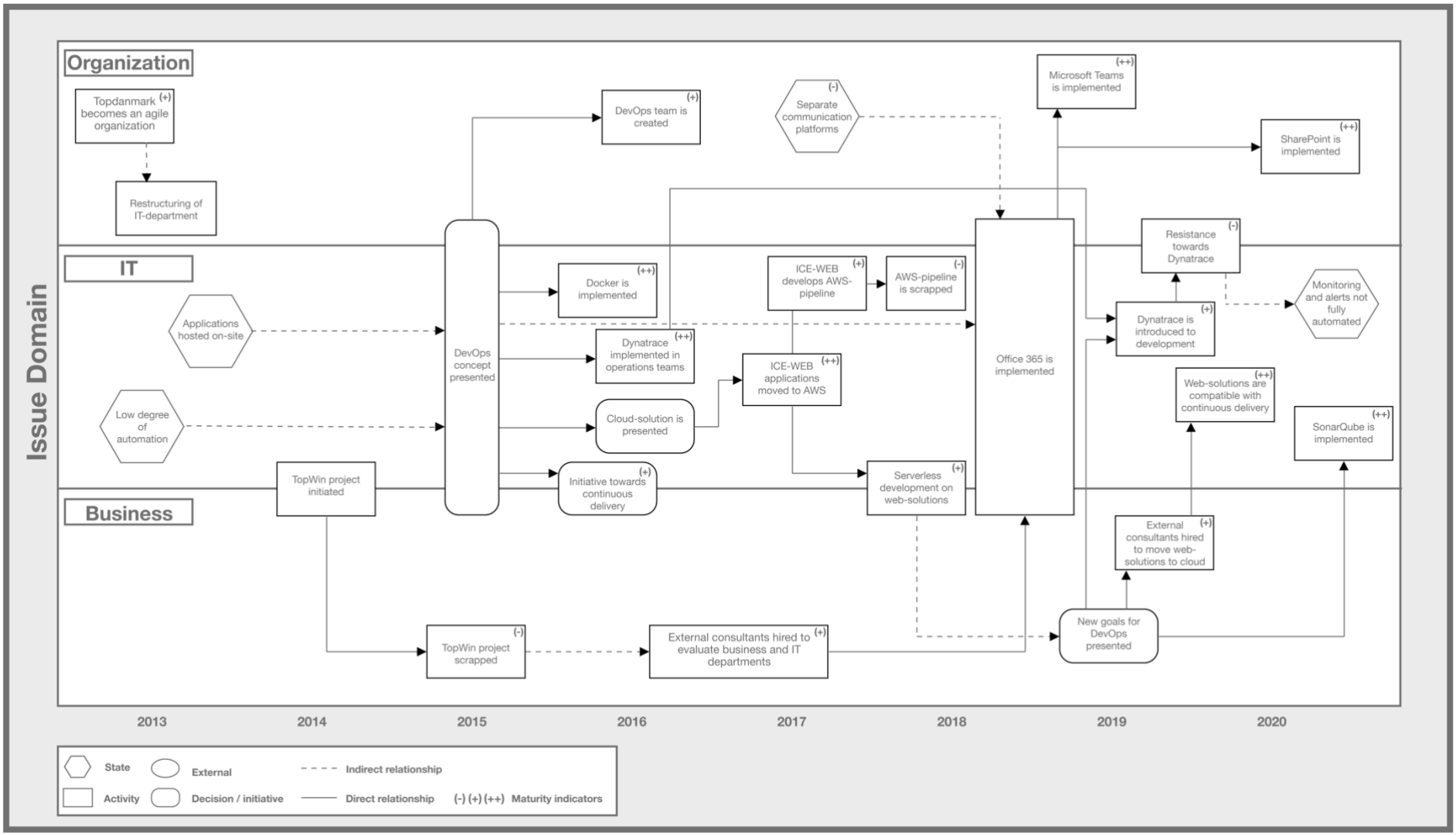


Figure 14: Visual map of Topdanmark

7 Comparative Analysis

7.1. Procedural Analysis

The evidence generated through the case studies shows that both organizations, prior to investigating DevOps, already fulfilled important agile principles, providing them with favorable starting positions for adopting DevOps. Each company has capitalized on this initial advantage and matured its DevOps approach significantly throughout the last five years. However, ProActive has managed to mature its organization in terms of both cultural and technical aspects, whereas Topdanmark has mainly improved in technical areas. Despite these outcomes, a consistent set of similarities in the events and activities performed by the companies emerged from the case studies.

Both companies have achieved the aspiration to move from a traditional release cycle to a continuous delivery setup. ProActive has fully transitioned to a *continuous delivery* approach with the change from being a conventional product to *IntraActive becoming a SaaS.* Likewise, Topdanmark has taken *initiatives towards continuous delivery* and deploy some applications daily, but the technical infrastructure still cripples the CD-process. The development and standardization of the technologies and tools that support the possibility of continuous development and delivery are found to have a significant impact on this transition. With ProActive's history as a Microsoft partner, they have adapted their software development to the tools developed by Microsoft. This strategy allows ProActive to continuously renew its software development while also following best practices, as Microsoft is one of the most prominent leaders and influencers in the industry. Topdanmark has sought to adopt a similar strategy by using tools issued by AWS, but due to the infrastructure issues, they do not yet receive the full value of this strategic approach. Thus, following industry leaders such as Microsoft and Amazon have piloted the way for a mature DevOps approach for the two case studies.

As a related effect, the companies also exhibit commonalities in terms of changed infrastructure. As part of the movement towards DevOps, both companies have become more cloud-based. ProActive has had their *on-site servers replaced with Azure* and uses exclusively cloud-based hosting for all their applications. Furthermore, ProActive has had its *technical infrastructure restructured* and has *developed its own API* to be able to deploy the desired SaaS delivery model. Topdanmark has, following a failed *TopWin project,* moved all *ICE-WEB applications to AWS.* As a measure to

discontinue the use of on-site hosting, Topdanmark hired *external consultants to help move additional web-solution to AWS* in the spring of 2019. Despite the success of shifting multiple web-applications to AWS, Topdanmark still has some on-site hosting today.

A consistent finding that emerged in both cases is the increased attention on product monitoring and optimized incident handling. For example, Topdanmark's introduction of *Dynatrace* initially in the operations teams and later in the development teams for better performance monitoring and faster error fixing. Topdanmark has significantly improved their monitoring, but still has cultural problems, such as developers not wanting to take additional responsibility, which blocks the progress. Also, the choice of monitoring tool is found to have a significant influence. Similar initiatives have been performed in ProActive in the form of *Application Insights* as well as the *implementation of telemetry*. With these measures, ProActive has moved their incident handling to a proactive approach. The use of Application Insights, was in the case of ProActive, an obvious choice as it is part of the Azure platform developed by Microsoft, hence the employees already knew the platform.

Conversely, Topdanmark has chosen to implement a third-party tool in the form of Dynatrace, which is compatible with their other tools and systems but has no specific connection. Thus, forcing the employees to add yet another platform to their workflows. Moreover, Dynatrace was initially chosen by the operations teams and then later implemented in the development teams, which has led to *resistance towards Dynatrace* by the developers as they do not consider the tool to be sufficient.

In terms of tests, the findings from the two cases emphasize increased attention on automated tests. The use of integration and unit tests is present in both companies as integrated parts in their build and release pipelines. However, the use of manual tests still exists in the two case companies. In ProActive, they make use of manual validation tests that must be completed before any feature or change can be moved to production. Likewise, Topdanmark has a dedicated tester connected to each development team that is responsible for testing any feature making its way to production. Topdanmark has, nevertheless, sought to increase the number of automated tests by introducing tools such as *SonarQube* for monitoring the percentage of code covered by automated tests.

The importance of streamlined communication is evident in the two case studies. Both companies have previously used various forms of communication platforms over the past few years, including

technologies such as Lync, IBM Connections, and Skype for Business. However, with the *implementation of Microsoft Teams*, communication among teams and employees has been improved substantially. The ability to ask questions or facilitate discussions with a broader crowd of employees enhanced the knowledge sharing process, especially in the case of ProActive that deems Microsoft Teams essential to their work and almost indispensable. In contrast, Topdanmark does use Microsoft Teams, which undeniably has improved their internal communication, but they are yet to centralize all the communication to one platform. The implantation of Microsoft Teams in Topdanmark has improved the alignment and knowledge sharing between teams, but there are still obstacles to surmount to heighten the communication in the organization.

Sufficient evidence from the two case studies shows that the need for structured and centralized documentation is essential to enhance DevOps maturity. Both companies have taken healthy initiatives to improve documentation. ProActive had before the research already implemented an intranet solution but supplemented this in 2018 by implementing a *central documentation platform* as well as setting up tools for *automatic API documentation.* At Topdanmark, a new *SharePoint* intranet was launched in 2020 to ensure a centralized approach to documentation. Furthermore, Topdanmark has also implemented the use of Gitbook for code documentation.

An interesting finding from the two case studies that correlate with previous literature was the necessity for cross-functional teams and merge roles. The companies have approached this in two completely different ways. ProActive has followed the theoretical approached and *merged the operations and development teams* into one. On the contrary, Topdanmark has created a new team, a so-called dedicated *DevOps team*, whose role is to act as a gap closer between the two traditional teams. In the case of ProActive, it is apparent that the merge of teams has moved them from a disjointed culture of “us” and “them” to a culture that exhibits a sense of co-responsibility, trust, and respect. In Topdanmark's case, it is different, as the newly created DevOps team has acted more like a functional link, providing the necessary tools for the continuous delivery process, rather than a cultural link between the development and operations teams. In fact, with the creation of the DevOps team, Topdanmark has introduced yet another entity with its own culture and perceptions towards software development. Topdanmark has acknowledged that moving the DevOps team to the development team would be a step in the right direction; however, they also believe that dedicated change management is required to encompass a thriving DevOps culture.

7.2. Challenges

As mentioned in the literature review section of this thesis, many different challenges have emerged and shown to have a direct impact on the success of a DevOps adoption. In the existing research, the most predominant challenges have been tied to the culture and mindset of employees, communication within the teams, fragmented planning activities, heterogeneous environments, and immature technical infrastructures, or the size of the investigated organizations (Elliot, 2014; Ghantous & Gill, 2017; Riungu-Kalliosaari et al., 2016). In this section of the analysis, we investigate and identify which challenges ProActive and Topdanmark have encountered in their DevOps processes. Simultaneously, we investigate whether any challenges have overlapped in the two case organizations during the last five years, and what the drivers and possible resolutions of the specific challenges are.

In both case organizations, fragmented planning activities have shown to be challenging. Development and operations often do not cooperate when new development projects are about to be launched, which means that the requirements, as well as the expectations for a given feature, might not align between the two teams. The neglected inclusion of all team members in the entire development process, from start to finish, has created problems in the organizations. While the existing literature focuses primarily on the cooperation between development and operations, both ProActive and Topdanmark have also encountered problems by not including certain employees outside of the development and operations teams early enough. Five years ago, ProActive encountered issues due to testers not being included early in the development process. By not being included in the initial development and planning of certain features or applications, the testers did not have any real insight into the processes and concept of the developed entity and could, therefore, not test it accordingly (Respondent 17, ProActive, 2015). ProActive has later made significant efforts towards including every team member in the entire development cycle of new features. By including everyone, ProActive has mitigated the risk of such an issue occurring again.

In Topdanmark, there are similar examples of fragmented planning activities. In both the previous and current data, developers have expressed issues connected to specific processes where they are not included sufficiently. In 2015, developers found it troubling not to be an essential part of the selection of user stories. According to an employee, business user stories were prioritized over technical user stories, due to the employees selecting the user stories having a business perspective

rather than a technical perspective. The developers were not included in the selection of user stories. They could, therefore, not explain individual technical user stories and get the business employees to *"acknowledge the value of them (technical user stories) and why they should not be deemphasized"* (Respondent 20, Topdanmark, 2015). Today, user stories are selected by the designated product owner of each team, and the developers of Topdanmark have not expressed any more extensive inclusion of them in the process. However, we did not perceive it to be an issue in the recent interviews.

A more significant issue raised by developers in Topdanmark in the new interviews was the exclusion in decisions about the selection of specific central tools. Seemingly, the operations teams have a "veto right" (Respondent 10, Topdanmark, 2020) on the selection of monitoring tools. In late 2015, *Dynatrace was introduced in Topdanmark* to be used by the operations teams for monitoring applications in the production environment. The intention was to introduce Dynatrace to the development team later as well and thereby provide them with the possibility of monitoring their developed applications. However, no developers were included in the decision to implement Dynatrace, and ostensibly the monitoring tool is not compatible with the AWS solution in the ICE-WEB team. Dynatrace can monitor the activity on the frontend solution but cannot trace calls downwards in the technical stack. The development team must implement other tools for monitoring of their web solutions running AWS, which sequentially results in increasing costs and complexity of the technical infrastructure. Had the developers been included before the implementation, a more suitable tool could have been chosen, mitigating the risk of encountering the present challenges.

A challenge in DevOps not thoroughly covered by the existing literature is external dependencies. Both ProActive and Topdanmark have encountered challenges in their DevOps processes derived from external dependencies. As ProActive is a Microsoft partner, IntraActive is based on SharePoint. External dependencies are introduced by having a product based on a third-party provider or external partner. In the case of ProActive, it has led to prior challenges in the configuration of deployments, since the configuration was dependent on a SharePoint crawl that ran in an unidentified interval (Respondent 18, ProActive, 2020). With the development of its own API, ProActive has taken steps towards bypassing the dependency of the SharePoint crawl. However, ProActive is still subject to difficulties when SharePoint implements changes. A change in a dependent module of SharePoint

could potentially have an impact on the IntraActive product and cause a restructuring of DevOps processes or the product itself.

In the case of Topdanmark, the primary challenges of external dependency are connected to regulatory compliance. Due to Topdanmark being an insurance company, it is subject to specific regulatory laws of the Danish state, such as special VAT-laws and requirements for data storage. For example, Topdanmark cannot deduct the VAT from purchases and therefore has to pay a 20% markup compared to companies in other industries, which could influence the potential purchase of DevOps tools and decisions concerning the technical infrastructure (Respondent 26, Topdanmark, 2015). Nevertheless, the purchase of tools and decisions about infrastructure is circumstantial, and one could argue that a 20% markup on everything should not affect the individual purchase since the company is also exempt from paying VAT to the Danish government. However, crucial regulatory compliance that Topdanmark is subject to is the audit requirement of segregation of duties. In deployment to production environments, it is a regulatory requirement for insurance companies to have segregation of duties (Respondent 26, Topdanmark, 2015; Respondent 12, Topdanmark, 2020). As such, developers of Topdanmark will not be able to push their developed code to the production environment without approval from a superior manager. Thereby, the regulatory compliance of Topdanmark introduces internal dependencies as well that have an impact on the deployment process. If there are not any superior managers available at a certain point in time, it has a significant impact on the continuous delivery process.

The findings of Walls (2013) and Riungu-Kalliosaari et al. (2016) show that organizational size is often a challenge in the adoption and use of DevOps technologies. They found that larger organizations are often hindered with DevOps due to their size and inability to react as fast as smaller organizations. However, in the investigation of ProActive and Topdanmark, we have not found any indication of challenges directly correlated with the size of the organizations. We have found a correlation between challenges and several factors that could be derived from the organizations' size but are not so explicitly. In our research, we find organizational structure, tenure of employees, and competences of employees pose a more prominent challenge than the size of the organization.

Both ProActive and Topdanmark have faced challenges in cross-team collaboration in the traditional organization structure of silo-divided teams. After ProActive's *development and operations teams*

were merged, the employees have not accentuated any of the identical challenges in interdisciplinary work. The risk of incurring challenges of interdependency and cross-team collaboration that existed has been mitigated as a result of combining the teams. Notably, challenges of interdependency and cross-team collaboration can still occur in ProActive's new organizational structure, but no challenges like those mentioned in the previous data have occurred since the merger. Most of the interviewed employees of Topdanmark are currently reporting challenges of cross-team collaboration and interdependency as a result of their organizational structure. The challenges Topdanmark is facing currently within interdisciplinary work are very much alike those from 2015. The consistent organizational structure could be a result of the large size of the organization, but is, according to the employees, a result of legacy systems and management decisions (Respondent 8, Topdanmark, 2020). In 2016, Topdanmark's *DevOps team was created*, but we have not found any significant impact on the organizational structure and mitigation of preexisting challenges thereof, other than adding an additional silo. There are still challenges of interdisciplinary processes within Topdanmark, but after *Microsoft Teams is implemented*, employees are reporting a significant reduction of complexity in cross-team collaboration as a result. Thus, both organizations have incurred challenges of interdependency and cross-team collaboration in a silo-divided organizational structure, and merging the development and operations teams is found to mitigate the risk of such challenges occurring. Furthermore, we found Microsoft Teams to reduce the complexity of cross-team collaboration, regardless of organizational structure.

Topdanmark has incurred challenges originating from the tenure and competences of employees. In both the previous and current data, employees of Topdanmark have emphasized the long tenure of employees and competences thereof as hindering the progression with DevOps. The extent to which employees accentuate this challenge varies, but as it is mentioned in the majority of the interviews, it should be investigated further. The tenure of many employees of Topdanmark exceeds two or more decades, which is much higher than ProActive. The long tenure of Topdanmark's employees has made the culture subject to more rigid mindsets and a lack of newer competences. According to the DevOps specialist, Topdanmark hired such employees to excel and function within the older systems, and with the change to newer technologies, their competencies are insufficient. *"People have not been hired with the purpose of knowing DevOps. There is a large gap of competences between the employees we have, and the people needed in a DevOps culture."* (Respondent 12, Topdanmark, 2020). Exactly what competencies employees with longer tenure in Topdanmark lack is not explicitly

disclosed in the interviews. However, following the assertion from a senior developer in ProActive (Respondent 2, ProActive, 2020), alongside the continuous improvement and availability of DevOps tools and practices, it becomes increasingly undemanding to use DevOps technologies. Correspondingly, requirements for technical competences of employees will, in the future, likely be lower. However, currently, competences of employees with longer tenure in Topdanmark is seen as a challenge in the organization's DevOps processes. The lack of competences is arguably also connected to the absence of any formal training of employees within the organization. We find the lack of training to be a relevant factor in the widening of the competence gap within Topdanmark.

The mindsets of employees in Topdanmark, and especially those of longer tenure, are being reported as rigid and a challenge in the adoption of a thriving DevOps culture by several employees (Respondent 7, Topdanmark, 2020; Respondent 24, Topdanmark, 2015). Many employees in Topdanmark are not open to and adequately motivated to undertake new processes in their workflow. Several developers will find it difficult to actively engage in operation tasks and vice versa (Respondent 7, Topdanmark, 2020). As the foundation of DevOps is to share responsibilities and tasks across areas of operations in the development and operations teams, we find the rigid mindset of some employees in Topdanmark as a challenge hereof.

In ProActive, rigid mindsets were existent five years ago but is not so currently. Thereby, ProActive has managed to change the mindsets of employees to suit a DevOps culture better. A possible explanation for the change of mindsets and the absence of challenges related to competences could be the *large turnover of employees in ProActive* in 2016. While the *large turnover of employees in ProActive* was challenging for the organization in many ways, as it replaced almost all employees in the IntraActive team, it could also have mitigated the risk of incurring challenges related to mindsets and competences of employees. As a completely new team was to be introduced in ProActive, the DevOps competencies of hired employees would be something that was considered.

Furthermore, as all employees were hired into a team starting “from scratch”, no existing workflows and processes of the individual employee had to be transitioned away from, thereby increasing the motivation of new employees and making it easier to adopt a thriving DevOps culture. Consequently, both Topdanmark and ProActive have been challenged by rigid mindsets, whereas ProActive has

successfully managed to overcome the challenge. Whether the resolution of the challenge in ProActive is directly corresponding with the introduction of a new team needs further research.

In contrast to the cultural challenges that can emerge in a DevOps setup, Topdanmark has incurred challenges due to their technical infrastructure. Topdanmark's mainframe that has existed since IT was introduced in the company and traditionally hosted all systems has proven to be a significant challenge in the adoption of DevOps. Topdanmark *"traditionally built everything to last for 40 years, which means that the turning radius is bigger than a small upcoming webshop"* (Interview X, TopD, old). Thereby, the technical dependencies of the organization are complicated to navigate when transitioning to a decoupled architecture with continuous elements. According to a developer, the legacy systems of Topdanmark pose a more significant challenge than the culture and people (Respondent 7, Topdanmark, 2020). The management of Topdanmark has been aware of the challenges engendered by the mainframe and tried to vanquish it with the TopWin project. However, due to economic reasons, the *TopWin project was scrapped*, leaving the technical infrastructure unchanged. Topdanmark has instead chosen a slower transition away from the traditional mainframe, but we deem the current technical infrastructure to have a noteworthy negative impact on Topdanmark's ability to deliver software continuously and integrate best practice DevOps processes.

7.3. Findings

The analysis of the two case organizations, both individually and compared, has provided us with a number of findings. The findings from the analysis are specific to the investigated organizations and will be presented as such. The generalizability of the findings is further discussed in section *8 Discussion*. The essential findings from the analysis are presented below (the findings are not hierarchized, or in any other way prioritized, according to importance):

- **Organizational size** is not found to have a direct impact on the DevOps approach of the organizations.
- **Organizational structure** is found to have an impact on the DevOps approach of the two organizations.
- **Tenure and competences of employees** are found to have an impact on the DevOps approach of the two organizations.
- A **DevOps mindset** amongst employees is necessary for a thriving DevOps culture.

- A **central communication and knowledge sharing platform** has thoroughly enhanced the communication and cross-team collaboration within the organizations.
- Both organizations have significantly enhanced automation and technical processes as a result of using **DevOps tools**.
- **Fragmented planning activities** have shown to be challenging for both organizations.
- Topdanmark's **technical infrastructure**, and dependencies thereof, have a significant impact on the ability to implement continuous processes.

Existing literature has found the **organizational size** to be a challenge in the implementation and use of DevOps technologies, stating that larger organizations are often hindered with DevOps due to their size and inability to react as fast as smaller organizations. However, in the investigation of ProActive and Topdanmark, we have not found any indication of challenges directly correlated with the size of the organizations. The organizational size of the investigated organizations can, however, have influenced several other factors that have shown to be challenging, but no direct link is found.

Both ProActive and Topdanmark have faced challenges of interdependency and cross-team collaboration in the traditional **organization structure** of silo-divided teams. Merging the development and operations teams is found to mitigate the risk of such challenges occurring. In Topdanmark, the consistent organizational structure could be a result of the large size of the organization but is inherently a result of legacy systems and management decisions.

Topdanmark has furthermore incurred challenges originating from the **tenure and competences of employees**. The long tenure of employees and competences thereof is hindering the progression with DevOps within the organization and has done so during the last five years. The tenure of many employees of Topdanmark exceeds two or more decades, which is much higher than ProActive. Long-tenured employees were hired to excel and function within the older systems, and with the change to newer technologies, their competencies are insufficient. The lack of competences is very likely also connected to the fact that there has not been introduced any formal training of employees in DevOps technologies. As such, the lack of training in DevOps technologies can be seen as a relevant factor in the widening of the competence gap within Topdanmark.

As the foundation of DevOps is to share responsibilities and tasks across areas of operations in the development and operations teams, a **DevOps mindset** is necessary. Rigid mindsets amongst employees are, therefore, a challenge in the adoption of a thriving DevOps culture. Both Topdanmark and ProActive have been challenged by rigid mindsets, whereas Topdanmark is still being negatively affected by the mindsets of employees, and especially long-tenured employees, while ProActive has successfully managed to overcome the challenge.

ProActive and Topdanmark have incurred challenges in communication and knowledge sharing in cross-team collaboration. Implementing Microsoft Teams as a **central communication and knowledge sharing platform** is found to have significantly reduced the complexity within communication and cross-team collaboration in both organizations.

We found that both organizations have made noteworthy progress in the product, process, and quality area of the maturity assessment during the last five years, primarily due to the utilization of **DevOps tools**. Automation and technical processes can be significantly enhanced with the wide availability of DevOps tools and relative simplicity in the implementation and use of such tools. There exists a challenge in choosing the right tools that will integrate with the technical foundation and processes of the organization seamlessly.

In both case organizations, **fragmented planning activities** have shown to be challenging. In immature DevOps setups, development and operations often do not cooperate when launching new development projects, which means that the requirements, as well as the expectations for a given feature, might not align between the two teams. It is likewise essential to incorporate affected teams in decisions about infrastructure and DevOps technologies. Furthermore, we found that it is not solely essential to include development and operations, but other teams or critical people, such as testers, as well.

Topdanmark has incurred challenges due to their **technical infrastructure** of legacy systems and dependencies thereof. We found that the technical infrastructure has a significant negative impact on the ability to deliver software continuously and integrate best practice DevOps processes.

8 Discussion

8.1. Organizational Size

The organizational size is not found to have impacted the adoption and maturity of DevOps within the two organizations. However, we acknowledge that the size of the organization could have indirectly influenced the organization's DevOps approach. Equal to the research of Nielsen et al.'s (2017), we found that the challenges faced by Topdanmark have been more noticeable compared to ProActive. Topdanmark has experienced persistent challenges throughout the last five years and is yet to overcome some key issues. This could be directly correlated to the higher number of employees involved in the software development, which means that a larger number of cultural understandings and perceptions exist. Generally, large organizations are more rigid and require formal structures and procedures to capitalize on change, whereas smaller organizations can easier adapt to shifts in the market (Lee & Xia, 2006). Thus, we argued that large organizations must increase their focus on cultural differences to achieve the desired DevOps culture.

Typically, the number of developers highly outweighs the number of operations people in software product organizations (Edwards, 2010). As such, it can be claimed that the size of an organization directly impacts its possibility to achieve the preferred cross-functional teams advocated for in DevOps research. At Topdanmark, there are a higher number of developers compared to operations employees (Respondent 9, Topdanmark, 2020; Respondent 12, Topdanmark, 2020), thereby corresponding to the findings of Edwards (2010). If Topdanmark were to merge the development and operations teams, each operations employee would still have to range over multiple development teams. Thus, it might mitigate cultural differences but the operations people would still not be fully integrated into the development teams. Though we found no direct correlation between organizational size and DevOps maturity, organizational size arguably has an impact on the structure of the organization.

8.2. Organizational Structure

In our research, we found the organizational structure of the investigated organizations to have an impact on the DevOps approach. In more detail, an organizational structure with divided development and operations teams incurred more challenges in interdependency and cross-team collaboration. On the contrary, we found that an organizational structure that consists of merged development and

operations teams mitigated the risks of incurring such challenges. As mentioned, the organizational size can arguably have an impact on the organizational structure, and questions can be raised about larger organizations' ability to break down silos and facilitate a merging of the development and operations teams.

In the case of Topdanmark, we found no correlation between the organization's size and organizational structure. Instead, we found that the inability to transition to a DevOps organizational structure, where development and operations are merged, could be explained by the technical architecture, management, and workflow consistency (i.e., "We have always done it like that"). Furthermore, there are numerous examples of much larger companies than Topdanmark that have successfully changed their organizational structure to facilitate DevOps, such as Google, Amazon, and Netflix (Diaz et al., 2018). These three corporate giants are all supported by an organizational structure of merged development and operations teams and have all been existent before DevOps was introduced and have therefore presumably transitioned from another organizational structure. Google, Amazon, and Netflix are arguable of entirely different characteristics than Topdanmark, which leaves questions of the effect of factors like industry, technical infrastructure, employees, and product on the organizational structure, rather than the size of the company. It is essential to underline that we do not disregard the size of an organization to have an impact on the organizational structure, but there exist more dominant factors concerning DevOps than size, as we have found in our research.

There are differences in the organizational structure of ProActive and Topdanmark, while both organizations are trying to facilitate a thriving DevOps culture. From our research, it is clear that ProActive has been more successful in achieving such. However, as mentioned, there is no "one-size-fits-all" for DevOps cultures, as every organization is different. Skelton & Pais' (2019) research on organizational structures, or "topologies", exemplifies some of the unique team structures that can, or cannot, facilitate a thriving DevOps culture. Skelton & Pais' (2019) research is based mainly on Conway's Law, stating that *"organizations which design systems… are constrained to produce designs which are copies of the communication structures of these organizations"*. Thus, further weakening the view of a single, identifiable DevOps culture. Skelton & Pais (2019) present two different types of organizational structures: *"DevOps team topologies"* and *"DevOps anti-types"*. Of these, DevOps team topologies are different, working organizational structures that support DevOps, and DevOps anti-types are "bad-practice" organizational structures.

The organizational structures of ProActive and Topdanmark are depicted in *Figure 15* and *Figure 16*, respectively. ProActive's organizational structure is what Skelton & Pais (2019) refer to as the "*fully embedded topology*". The fully embedded topology is reminiscent of minimal separation between development and operations, and is, according to Skelton & Pais (2019), ideal for organizations with a single main web-based product or service. As such, the fully embedded topology is a particularly good fit for ProActive.

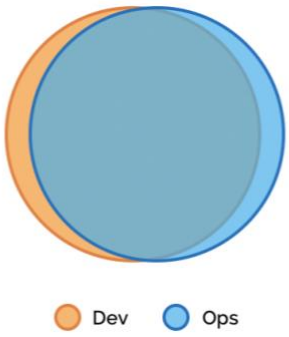


Figure 15: The fully embedded topology (Skelton & Pais, 2019)

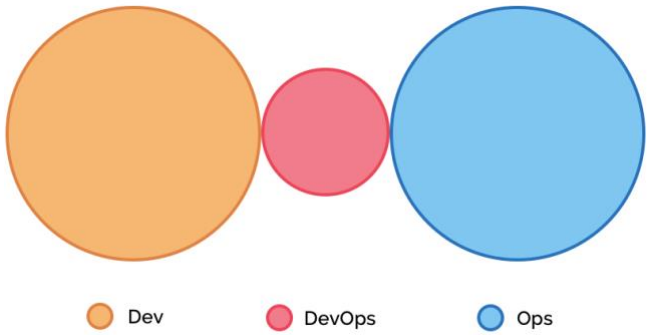


Figure 16: The DevOps team silo (Skelton & Pais, 2019)

For Topdanmark, however, we found that the creation of the DevOps team did not have a significant impact on the organizational structure and DevOps approach, other than creating another silo internally. In the research of Skelton & Pais (2019), the organizational structure of having a development, operations, and DevOps team can either be seen as a DevOps team topology or a DevOps anti-type, mainly depending on the intent and longevity of the structure. The creation of a DevOps team can be a rewarding topology if the goal is to bring the development and operation teams closer together towards a fully embedded or more collaborative topology and eventually make itself obsolete. However, this organizational structure must be a temporary structure with the longevity of a maximum of 1 or 2 years. Otherwise, it becomes a DevOps anti-type, and merely another silo in the organizational structure that broadens the distance between the development and operations teams. In the case of Topdanmark, their organizational structure can, by now, arguably be identified as a DevOps anti-type. In our research, we found Topdanmark's current organizational structure to be recognized as a factor that creates many difficulties regarding cross-team collaboration. Following Skelton & Pais' (2019) research, Topdanmark should have made the DevOps team obsolete several years ago to avoid introducing additional challenges in their DevOps approach. According to the findings of Diaz et al. (2018), the strategic decision to create a separate DevOps team instead of

merging development and operations is often due to financial limitations. To simplify, "*Most companies cannot afford to extend DevOps to all their development teams, so they are organizing their people around teams that are supported by a DevOps team"* (Diaz et al., 2018). As Topdanmark is relatively large, the financial costs of merging the development and operations teams would arguably be very high, and substantially higher than in a smaller organization such as ProActive. While the financial costs are not the reason behind the current organizational structure, the rising cost within a large organization like Topdanmark is acknowledged.

As such, we argue that the organizational structure and DevOps topology have a tremendous impact on the organizational ability to adopt a thriving DevOps culture. When comparing ProActive and Topdanmark, it is clear that ProActive has incurred significantly fewer difficulties in their DevOps approach, after the merging of its development and operations teams. We further argue that organizational size does not impact the organizational structure significantly. However, we recognize the possible rising financial costs of breaking down organizational silos that positively correlate with organizational size. Based on Skelton & Pais' (2019) research, we recommend that Topdanmark either rapidly make an effort towards making the DevOps team obsolete, and merge the development and operations teams, or choose another topology and incorporate it in its DevOps strategy, in order to mitigate the risk of incurring additional challenges in cross-team collaboration.

8.3. Tools and Infrastructure

In our research, we incorporate two main findings that appertain to the technical aspect of DevOps. These are concerning the impact of DevOps tools and technical infrastructure on the DevOps adoption of an organization. Both ProActive and Topdanmark have made tremendous progress in their technical DevOps processes due to the implementation and utilization of various DevOps tools. However, in the case of Topdanmark, we found that the tightly coupled technical infrastructure introduced setbacks to the DevOps processes of the organization.

When comparing the current maturity assessments of ProActive and Topdanmark, it becomes evident that both organizations have made significant progress within the Product, Process, and Quality section. As most areas within the Product, Process, and Quality section can be condensed to automation processes, it is likely as a result of the implementation and utilization of DevOps tools. This aspect aligns with the statement of a senior developer in ProActive: *"DevOps is so simple in its*

form, and the tooling is so advanced today that you do not really have to do anything […]" (Respondent 2, ProActive, 2020). The wide range and specificity of DevOps tools can be due to the "*Cambrian explosion of DevOps tools",* as described by Kersten (2018). The Cambrian explosion of DevOps tools is manifested in the rising market of DevOps tools that has been formed to fill the gap created by the waterfall model's displacement (Kersten, 2018). The term covers the growing number of DevOps tool vendors that each are trying to provide a repository or automation layer for a segment of the software value stream. The DevOps toolchain's specialization is following the growing complexity of software development, which can explain the full range of DevOps tools and the advancement of such tools. Therefore, organizations arguably stand before a decision to choose the right tools between the plentiful supply that will integrate seamlessly into the processes and technical infrastructure of the organization. However, the effort and costs required for implementing the tools are decreasing and are mainly dependent on the technical infrastructure of the organization.

We found the use and implementation of tools to be significantly improved in the decoupled architecture of ProActive. With a more modularized architecture, it is possible to upgrade smaller parts of the system independently, and it introduces shorter wait times for build, test, and deployment results (Smeds, Nybom, & Porres, 2015). On the contrary, Topdanmark incurred challenges due to the monolithic architecture of its mainframe, making it tremendously more difficult to implement tools and utilize them to the intended extent. However, focusing on the ICE-WEB team within Topdanmark that has a more decoupled architecture, without dependency on the mainframe, it is possible to see the difference, as the ICE-WEB team has introduced a lot more automation. As such, Topdanmark is not getting the most out of their DevOps approach, as their legacy systems cripple them.

In 2014, the TopWin project was initiated to transition away from the legacy systems, but the project was scrapped. Retrospectively it might have been the correct decision. Following the research of Kersten (2018) and Elliot (2014), transitioning away from legacy systems or trying to adjust them to meet DevOps practices can be tremendously expensive and dangerous. Elliot (2014) has found that IT organizations that have tried to custom adjust their technical infrastructure to meet DevOps practices have a failure rate of 80%, which advocates for the critical requirement of replacing or adding new tools to the existing infrastructure. Following the research of Kersten (2018) and Elliot (2014), if Topdanmark once again makes a strategic decision to improve its position with the technical

infrastructure to meet DevOps practices, the management should aim to disintegrate the monolithic infrastructure and replace it with a decoupled counterpart. However, restructuring the technical infrastructure entirely will arguably be tremendously costly due to the numerous dependencies, and the decision to do so is perhaps not economically feasible. Furthermore, the aspect concerning the need of Topdanmark to progress their DevOps approach and maturity level, and if a replacement of the technical infrastructure is rationalized, will be further discussed below.

8.4. Communication

Through the investigation of ProActive and Topdanmark, we found that a central communication and knowledge sharing platform has significantly enhanced the communication and cross-team collaboration within the two organizations. Both organizations have implemented Microsoft Teams, which has shown to be a catalyst for streamlining communication and knowledge sharing through its channel structure and file support integration. This implementation aligns with extant research on communication tools within software development, where findings show a staggering increase in the use of chat-based systems such as Microsoft Teams (Silva, Gilson, & Galster, 2019; Alkadhi et al., 2017). According to Alkadhi et al. (2017), the increase in chat systems' popularity is due to the numerous decisions that development teams make throughout the software lifecycle. Developers and operations require team members' opinions as a continuous flow of information is needed to constitute a rationale for the many decisions and actions performed (Alkadhi et al., 2017). Similar to other IT systems, the use of chat-based communication requires the employees' acceptance to achieve the value it generates (Silva et al., 2019).

Though the two organizations have experienced similar value from the implementation of a central communication and knowledge sharing platform, the use of such is more integrated into the daily work of ProActive compared to Topdanmark. In the case of Topdanmark, we found no correlation between communication practices and organizational size. However, the less integrated use of Microsoft Teams in Topdanmark can arguably be connected to the number of employees and the need for additional acceptances towards the system. In contrast, ProActive's fast adoption of Microsoft Teams can be rooted in the attachment and partnership with Microsoft as well as the origins of their product, an intranet. ProActive is arguably experts in the field of communication and collaboration systems and even more so when it comes to products developed by Microsoft. Thus, the acceptance

and integration of Microsoft Teams are achieved more rapidly and seamlessly in an environment like the one of ProActive.

8.5. Tenure, Competences, and Mindset

Topdanmark has experienced challenges originating from the long tenure and lack of DevOps competencies of employees, which hinders them from progressing with DevOps. Barriers will emerge when a transformation, such as adopting DevOps, is performed. Changing the way of thinking and acting will most likely meet resistance, and the main problem is often related to the culture and mindset of the employees (Anderson & Anderson, 2011). In Dam, Oreg, & Schyns's (2008) research on change management, they found a positive correlation between organizational tenure and change resistance. Organizational changes are often tied to changes in the employees' daily work, which is why long-tenured employees are less inclined to accept changes in their work situations, as it must be assumed that they are satisfied with their current job situation. Thus, long-tenured employees may exhibit greater resistance to the change. Furthermore, long-tenured employees are expected to have invested in their jobs by acquiring skills or knowledge, which in turn can be a factor of resistance due to the fear of diminishing these investments (Dam et al., 2008). This rationale agrees with our findings at Topdanmark, where several developers and operations, whose employment exceeds multiple decades, have exhibited resistance towards DevOps, as they do not feel the need to undertake operations tasks or vice versa.

DevOps does not have any specified methodologies; hence companies are obliged to develop competences and practices for DevOps continuously. The on-going development of competences and practices is accentuated as a challenging process, and is, in ProActive and Topdanmark, characterized by retrospectives and a *"learn-by-doing"* approach (Respondent 9, Topdanmark, 2020). Consequently, employees must receive gradual training in line with the development of the company's DevOps approach (Diaz et al., 2018). In the case of Topdanmark, it is unknown exactly which DevOps competences the employees are lacking. However, the company has not tried to eradicate this challenge by conducting formal training sessions. This could arguably be connected to the company's divergent focus on DevOps. Topdanmark has over the last five years presented DevOps as a new initiative to the employees on several occasions, which indicates a lack of strategic alignment. The management's fluctuation on DevOps was manifested in the interviews with the employees as we experienced that multiple employees did not know what DevOps is or what the

purpose of DevOps in Topdanmark is. As such, it can be argued that a lack of a transparent strategy has had an impact on the employees' interest in the concept, and as a consequence, the employees have not felt compelled to learn the new contexts or change their work habits. To address this problem, Topdanmark should establish transparency in its IT strategy as well as align DevOps expectations between employees and management.

In the research of Diaz et al. (2018), it was found that convincing people of the purposes and values achieved from building strong relationships between development and operations are challenging. Some of the developers in Topdanmark are, as described by one employee: "*developers with a capital D*" (Respondent 7, Topdanmark, 2020), and are therefore not interested in undertaking operational tasks or share development tasks with operations. Whether this is due to misgivings towards losing their jobs or just general disinterest in DevOps needs further research.

Another concern could be the fear of being overburdened with additional responsibilities associated with operations or vice versa. Taking on additional responsibilities could lead to the loss of time to focus on their preferred role and, thus, affect their productivity (Smeds et al., 2015). These concerns contribute to a mindset and culture of "if it is not broken, do not fix it*"*, where employees' willingness to share, communicate, and collaborate is reasonably narrow. Elliot (2014) found that cultural inhibitors that prevent the establishment of these cross-functional relationships were the most prominent DevOps challenge. A staggering 56.7% of the investigated cases in Elliot's (2014) research reported cultural inhibitors as the biggest challenge for DevOps adoption. Although the tooling in DevOps has become so technologically advanced and can assist organizations in the DevOps transitions, it cannot resolve cultural impediments. As such, organizations like Topdanmark need to surmount these cultural barriers to achieve a successful DevOps approach.

8.6. Fragmented Planning Activities

Elliot (2014) identifies fragmented processes as one of the most prominent DevOps challenges. In the investigation of ProActive and Topdanmark, we found a similar result as both companies had experienced challenges associated with fragmented planning activities. Finding the right balance between what employees to include in the planning of new features was shown to be a challenging task, as it is costly to include every involved employee. On the contrary, excluding specific roles for the initial planning can lead to obstacles later in the software development cycle, as critical

perspectives from either a development or operational view can be missing. ProActive decided to include every team member in the planning of new features by hosting several workshops where both business and technology-related topics are discussed (Respondent 4, ProActive, 2020). At Topdanmark, they are yet to figure out how to eradicate this issue as the planning process still follows a random approach where individual employees are included upon availability (Respondent 8, Topdanmark, 2020). The aspect of including developers in the decisions of implementing specific tools will also have to be addressed by Topdanmark, as it has resulted in challenges and a further divide between the teams. Thus, planning activities should include a collection of employees that represent the development, operations, and business side of the organization to mitigate the risk of incurring related problems in the future.

8.7. The Perceived Value of DevOps Maturity

ProActive and Topdanmark have significantly improved their DevOps approach, thereby reaching a higher level of DevOps maturity. The general presumption is that there is a positive correlation between maturity and value, i.e., a higher level of maturity leads to an increase in value. However, it is essential to discuss the validity of this presumption and the boundaries of the expected value, as these might differ across organizations. Furthermore, the aspect of becoming too DevOps mature, where the marginal value gained from a higher level of maturity is lower than the costs, is not covered by existing literature. Thus, the aspect of becoming too DevOps mature is further discussed.

With the introduction of DevOps, and the maturity progress during the last five years, both ProActive and Topdanmark have experienced an increase of value in several overlapping areas. Some of these include increased organizational agility (e.g., faster time to market) and a significant reduction of deployment costs. Topdanmark's ICE-WEB team exhibits considerable reductions in deployment time, from weeks to minutes, as previously fixed release schedules have been replaced by the ability to deploy continuously. One employee from Topdanmark describes the improvement as:

> "*In our Top-Up process, we had a minimum time-to-market called 14 days. There you really had to be sharp if you could do it in 14 days [...] with the CD-process today, you have a process time consisting of your build time and your deployment time, so about 5+5 minutes*." (Respondent 12, Topdanmark, 2020).

ProActive has experienced similar gained value from increased organizational agility, thereby coinciding with the research on DevOps value from the existing literature. The introduction of continuous delivery in ProActive and the ICE-WEB team of Topdanmark has reduced the deployment costs significantly in their DevOps adoption. This aligns Gruver & Mouser's (2015) research, who found that the implementation of DevOps and agile principles will decrease the marginal cost of delivering a new feature to almost nothing. As such, the adoption of DevOps has shown to provide organizations with measurable benefits in the form of reduced cost, in addition to the intangible benefits presented in the existent literature.

Following the research of Dingsøyr and Lassenius (2016), the introduction of DevOps in organizations has a positive effect on the organization's brand. Both ProActive and Topdanmark have experienced the introduction of DevOps to have a positive impact on their organizational brand. However, we did not find the recognition of the improved brand to come from the customers, as it is described in the literature. Instead, ProActive and Topdanmark accentuated the positive effect of having DevOps on the attraction and acquisition of talent. As such, the improvement on the organizational brand that DevOps can provide is not seen as a significant value, when focusing on customers, but instead realized when the organizations are hiring new employees. We argue that the introduction of DevOps does not improve the organizational brand towards customers, as the average customer does not know, or perhaps care, how software is developed and delivered by the organizations', as long as it reaches the customer. Introducing DevOps in organizations could improve customer satisfaction, but not directly the customers' perception of the organization's brand. The impact of having DevOps within the organization on attracting employees can be tremendously difficult to measure, as it is purely subjective to the new employee. Also, we argued that a higher level of DevOps maturity will not significantly impact the attraction of employees, as it is perceived to be the presence of DevOps and not the level of DevOps maturity that attracts possible employees.

Following the brief discussion of realized value within the case organizations, questions can be raised concerning the possibility of deriving significant additional value from DevOps and whether further progress with DevOps is economically feasible. It can be argued whether a higher level of maturity directly correlates with an increase in value given that all organizations do not benefit equally from progressing with DevOps. The existing research on DevOps maturity models does not cover the aspect of becoming too mature, and it is solely mentioned in the future research of the existing

literature. Consequently, in answering the question of whether organizations can become too DevOps mature, we cannot draw parallels to existing research, and we will therefore have to base it on the cases of ProActive and Topdanmark that we have investigated.

As previously stated, Topdanmark's technical infrastructure hinders them from fully integrating continuous delivery to all parts of their software development cycle. From a theoretical perspective, especially, this is seen as a tremendous challenge that prohibits Topdanmark from further progressing with DevOps. However, it is essential to consider the individual case and rationalize the existing research to practice. While the technical infrastructure of Topdanmark indeed is a significant challenge in progressing with DevOps, it is essential to consider if it is feasible for the organization to become more mature, and transition away from their technical infrastructure, as the costs concerning that transition, are momentous (as seen in the scrapped TopWin project).

As such, we adopt the view of not only what hinders organizations' in DevOps and if they *can* mature, but also whether they *should* mature. When working with maturity models, we find it essential to argue the blatant assumption of "more is better" and discuss the maturity levels of the case organizations in specific relation to their situation. When looking at the two case organizations, it is clear that they are of vastly different characteristics, indicating that they might differ individually concerning their optimal level of maturity. ProActive and Topdanmark have significant differences in industry, product, and customers, arguably leading to a varying need for the ability to provide software continuously.

ProActive's product is an intranet, and such an application intends to provide the users with a platform for collaboration, communication, and knowledge sharing on a daily basis. For example, Copenhagen Business School uses the IntraActive product, where 20.000+ students and faculty rely on the intranet to get information and interact with the university daily (IntraActive, 2020). When so many users potentially interact with a system every day, it is arguable that the need for continuous releases of features and bugfixes is higher than in a system with fewer users and less repetitive usage. Furthermore, the need for a high level of DevOps maturity increases, as SharePoint is one of ProActive's external dependencies in delivering software. This is because SharePoint is a SaaS, and since IntraActive is built on SharePoint, ProActive has to continually revisit the development of their product to ensure compatibility between SharePoint and IntraActive.

With a higher need for continuous processes and a more mature DevOps approach, it is likely advantageous for ProActive to continuously improve their DevOps approach and progress their maturity to a prodigious level to meet the expectations, their product demands. Currently, ProActive still lacks automation in certain areas that have been highlighted by the developers, such as onboarding of customers, but the mindset within ProActive is to continuously improve and *"always become better"* (Respondent 2, ProActive, 2020). Therefore, we argue that ProActive can still gain value from improving their DevOps processes, as their technical infrastructure allows for it, their product and customers require it, and the marginal benefits arguably outweigh the marginal costs.

In the case of Topdanmark, the circumstances are quite different. Contrary to ProActive, Topdanmark delivers applications within the insurance industry, where customers do not have the same requirements and use patterns. It is very improbable that customers in the insurance industry are using the applications on a daily or weekly basis, and therefore, the need for system changes and features become less severe than in the case of ProActive. Besides, several of Topdanmark's applications rely on dependencies to their mainframe, and it will not be possible to deploy any full-stack changes in the software continuously as a result of the monthly mainframe deployments. Topdanmark's applications arguably do not constitute a high level of DevOps maturity due to their customers and technical infrastructure capabilities. However, Topdanmark could benefit from maturing DevOps on an organizational and cultural level to achieve higher development quality.

Following the analysis and subsequent discussion on DevOps maturity within the two case organizations, we have found the answer to becoming too DevOps mature to be purely circumstantial. From our research, an organization cannot reach a point of DevOps maturity where it is disparaging of efficiency or damaging to the organization. We argue that organizations can reach a level of DevOps maturity, where it is no longer economically feasible for them to progress in maturity. However, it depends significantly on the context of the individual organization, and the organization's circumstances concerning several factors such as industry, product(s), customers, and technical capabilities.

8.8. Implications for Theory

Our research represents major contributions to the existing theory and practice on DevOps maturity and DevOps in general. We have identified multiple activities, challenges, and aspects for consideration of any organization that seeks to adopt or progress its DevOps approach successfully. As such, our research contributes to the existing knowledge base by expanding the knowledge found in the existing research.

In summary, our research highlights the importance of structuring development and operations teams to facilitate cross-functional collaboration. In agreement with the existing literature, we found that a merge of teams is the optimal choice due to the mitigation of interdependencies. We further enhance the existing literature by emphasizing the need for encompassing a healthy DevOps culture where development and operations possess a positive mindset towards DevOps. Furthermore, our research acknowledges previous findings that highlight the significant influence of technical infrastructures and the rise of DevOps tools on the adoption and progression of DevOps maturity. We also recognized fragmented planning activities as an obstacle that can impact the development and DevOps approach of software organizations, similar to the extant literature. However, contrary to existing literature, we found the organizational size not to impact the DevOps maturity of an organization.

Our research presents interesting new findings, as well. We found that optimal communication and knowledge sharing within DevOps teams can be established through a centralized platform such as Microsoft Teams. The use of such a platform enables a strong relationship between development and operations. Moreover, we identified tenure and competences to be a notable influence in the adoption of DevOps. The impact of tenure and lack of competences was not yet covered in the existing literature on DevOps. However, our findings do align with the existing literature on change management. Lastly, we found that circumstantial parameters, such as industry and type of product, affect the necessity for DevOps maturity as our research found that the marginal benefit derived from DevOps, in some cases, is exceeded by the associated marginal cost.

8.9. Implications for Practice

In software product organizations, adopting DevOps is challenging. It requires a fit between the development and operation teams, which is highly dependent on the success of the integration of the teams, and the ability of team members to share and take on additional tasks and responsibilities.

DevOps may also require substantial investments in collaboration, knowledge sharing, technical infrastructure, and organizational culture.

With the growing adoption of DevOps, organizations are wondering what the value proposition of DevOps is and how they can harness it. Our study suggests DevOps' primary value proposition is organizational agility, and more specifically, increased efficiency in the development, deployment, monitoring, and incident handling of a software product. The mere introduction of DevOps may, however, not necessarily produce greater organizational agility. We see that our research has practical implications in three aspects.

Firstly, our analysis provides a detailed view of the case organizations' processes and decisions concerning DevOps, and the progression of their level of maturity, during the last five years. ProActive and Topdanmark can utilize our analysis as an assessment of their current state within DevOps, and as a reflecting tool that can drive future strategic decisions. Furthermore, our research identifies several challenges and shortcomings in the case organizations' DevOps approaches that are hindering, or need more attention, for the organizations to progress further with DevOps.

Secondly, our research and findings provide great insight for other software product organizations looking to adopt DevOps or mature their DevOps approach. Our analysis studied the cause and effect relationship of events in DevOps that ultimately resulted in challenges or success for the investigated case organizations. Thus, our findings consolidate the prior works and contribute by showing how problems and solutions are related to the processes of DevOps.

Lastly, we discuss the presence of an optimal level of DevOps maturity, where it is no longer economically feasible for organizations to progress in maturity. As prior work not thoroughly covered the circumstantial element of maturity, our discussion concerning the need for progressing DevOps maturity can act as input in strategic decisions of organizations faced with similar predicaments.

8.10. Limitations

In the process of conducting research, we have found several limitations that merit consideration. Here, we present the limitations that we have deemed to be most significant to our research. We do

not disregard the presence of unmentioned limitations, but only those with striking importance for our research are included in the following.

Working with two distinct cases in this paired-case study has provided another dimension to the analysis. While the two organizations have different characteristics (i.e., age, size, industry, organizational structure), they have found themselves in a similar situation of adopting DevOps across their organizations. Due to both organizations adopting DevOps, and within the same timeline, it has made them possible for comparison. The unique characteristics of ProActive and Topdanmark can be seen as a limitation and a benefit. Because of the differences in the two cases, questions can be raised about the appropriate comparison. Are the adoption and use of DevOps enough for a direct comparison, or should the cases carry more similarities for an adequate comparative analysis? Contrariwise, the differences of the case organizations be a benefit in the comparative analysis. In recognition of differences in size, industry, organizational structure, and other variables, the research can be constructed to incorporate the differences, and investigate the possible impact of such factors on an identical situation. As such, the paired-case comparison of this thesis highlights possible patterns and differences in the adoption of DevOps that is applicable on a general or specific level. However, careful consideration should be taken when applying the findings of this research to other cases.

The application of the results on a general level can be argued due to the sample size of the comparative analysis. The reliability of the findings in the analysis is increased compared to a single-case study, but the results are still circumstantial (Andersen, 2014). To increase the reliability of the findings, we could have included more case organizations in the analysis. However, while incorporating more case organizations would improve reliability, it would also require vastly higher amounts of time and resources. Furthermore, if more case organizations were to be included, it would need previous data from the past five years, which would be very difficult to obtain and increase the number of resources accordingly. As such, we recognize that the results of the analysis are not to be generalized insouciantly but instead contributed to the existing knowledge base within DevOps research and qualitative longitudinal research.

The data used in this thesis consists of both secondary and primary data. Both data sources are generated through the data collection technique of semi-structured interviews. With this method,

certain biases can occur and influence the data. When conducting interviews, social desirability should always be considered as respondents seek to present themselves in the most socially desired way (Barriball & White, 1994). For example, developers could be assumed to portray their efforts and work in the best possible way. Thus, social desirability can contribute to the weakening of reliability, as data affected by this is not methodologically transparent. The risk of respondent misinterpretation is another factor relevant to this thesis, as the consistency of stimulus in semi-structured interviews is dependent on the interviewer's ability to convey equivalence of meaning to maintain comparability (Barriball & White, 1994). The semi-structured technique allows the interviewer to rephrase and change the wording if it benefits the respondents' understanding of the question. This type of freedom opens the risk for differences in answers due to misinterpretations of the questions (Andersen, 2014).

Moreover, the respondents from the secondary data were anonymized, and we did not have the opportunity to ensure that we interviewed the same employees. However, as a measure to overcome this issue, respondents for the primary data were selected based on the roles included in Nielsen et al.'s (2017) research to reach the highest level of comparability. Thus, comparability between the two data sets is slightly reduced, but it is not considered to have a significant impact on the results.

The retrospective interviews of this thesis were centered around events and activities that occurred in between the two data points (2015 and 2020). We asked the two respondents to recall and describe events that influenced their software development cycle and DevOps approach. There is a likelihood that the two respondents have forgotten essential events or could not recall exact details of occurred events.

8.11. Future Research

The limitations section showed that the research came with some limitations, which form input to future research. First, as this thesis is based on a paired-case study, and thereby a sample size of only two organizations, it has an impact on the reliability of our findings. Especially those of our findings that contradict existing research or contribute new knowledge can benefit from further investigation. The impact of factors such as organizational size and structure as well as mindset, competences, and tenure of employees still require more research to determine. Notably, these factors do not need the boundaries of a longitudinal research design to be further investigated. Second, our research suggests

that more research on the effect of industry on DevOps is needed. We propose that the industry of organizations contribute to a circumstantial element that influences the DevOps adoption of organizations and their optimal level of maturity and is an aspect that calls for additional research. Third, our research pioneers the discussion concerning the possibility for an organization to become too DevOps mature. As this aspect is not covered by existing literature, it requires more research with new empirical entities to be justified. Concurrently, such research can further investigate the existence of an optimal level of DevOps maturity for different organizations. Fourth and last, more research and empirical work are vitally needed to practice and validate the use of DevOps maturity models. As the knowledge base of DevOps maturity is rather scarce, is merits additional research to be justified as well as to increase the generalizability and applicability of DevOps maturity models.

9 Conclusion

The constantly changing business- and technical requirements for IT products have caused a paradigm change towards continuous delivery that allows organizations to release code faster to the market. With this paradigm shift, the DevOps software development philosophy has emerged. While DevOps reside within the world of software development and is directly coupled to technology, it is first and foremost a question of culture. DevOps advocates for breaking down the traditional walls of development and operations to align incentives through culture, automation, monitoring, and test. Despite the increased attention on DevOps by practitioners, research on the adoption and maturing of DevOps is still rather scarce. As such, this thesis sought to address how organizations can mature their DevOps approach.

We found several drivers and capabilities that can progress organizations' DevOps maturity. In line with other studies, we found that the merging of the two traditional teams of development and operations can accelerate a DevOps culture and improve organizations' DevOps approach. In conjunction, organizations that are unable to merge Dev and Ops teams can create a dedicated DevOps task force to facilitate the maturing process. However, such task force should act as a catalyst and not a permanent solution.

Through the investigation of two Danish organizations, we found that encompassing a thriving DevOps culture is essential for an organization's approach to DevOps. Without a positive mindset towards cross-functional collaboration and the willingness to undertake tasks and responsibility that reside outside an employee's traditional field of work, the philosophy of DevOps cannot function effectively. We found the introduction of a central platform for communication and knowledge sharing to improve cross-functional collaboration and foster strong relationships between Dev and Ops.

The number of DevOps tools is increasing, and the availability of specialized tools can provide organizations with an advantage in DevOps. Specifically, the automation of technical processes can be aided by the utilization of tools, and thereby assist in progressing the maturity of technical aspects within DevOps. Thus, organizations stand before a decision of choosing the right tools that will integrate seamlessly into their technical infrastructure.

Furthermore, we identified factors that hinder organizations from maturing DevOps. First, we observed tenure and competences of employees to be highly influential on DevOps maturity. Long-tenured employees were less prone to embrace the organizational changes triggered by DevOps and thereby less interested in acquiring the necessary competences for a successful DevOps approach. Second, we found the exclusion of certain roles in planning activities of organizations to be a challenge in the development processes and implementation of DevOps technologies. Third, we conclude that the technical infrastructure of an organization significantly influences the adoption of DevOps. We found that a decoupled infrastructure is superior to a monolithic infrastructure in maturing DevOps.

As part of our research, we investigated the possibility for organizations to become too DevOps mature. We take a critical stand towards the blatant assumption of maturity models where "more is better" and pioneer the discussion of whether organizations should seek to further mature their DevOps approach. We conclude organizations cannot ascend to a level of DevOps maturity that is disparaging of efficiency or damaging to the organization. We assess that organizations can reach a point where it is no longer economically feasible to progress their DevOps maturity. This assessment is bound by circumstantial parameters concerning the individual organization, such as industry, product, customer, and technical capabilities.

By concentrating on the identified drivers and capabilities from this research, organizations can successfully mature their DevOps approach. Acknowledging the challenges found and applying focus to comprehend these can provide organizations with a stronger position for maturing DevOps. Lastly, organizations should critically evaluate their surrounding circumstances and assess the value derived from increased effort towards DevOps maturity.

10 Bibliography

Aiello, B., & Sachs, L. (2016). *Agile Application Lifecycle Management: Using DevOps to Drive Process Improvement.* Addison-Wesley Professional.

Akshaya, H. L., Vidya, J., & Veena, K. (2015). A basic introduction to devops tools. *International Journal of Computer Science & Information Technologies, 3*(6), 5-6.

Alkadhi, R., Lata, T., Guzmany, E., & Bruegge, B. (2017, May). Rationale in development chat messages: an exploratory study. *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, 436-446.

Andersen, I. (2014). *Den skinbarlige virkelighed: Vidensproduktion i samfundsvidenskaberne* (2 ed., Vol. 5). Samfundslitteratur.

Anderson, D., & Anderson, L. A. (2011). *Nøglen til ledelse af forandring: strategier for bevidst forandringslederskab* (Vol. 1). Gyldendal Business.

Barriball, L., & While, A. (1994). Collecting Data Using a Semi-Structured Interview: A Discussion Paper. *Journal of Advanced Nursing*(19), 328-335.

Becker, J., Knackstedt, R., & Pöppelbuß, J. (2009). Developing maturity models for IT management. *Business & Informations systems engineering, 1*(3), 213-222.

Creswell, J. W., & Clark, V. L. (2011). *Designing and Conducting Mixed Methods Research.* SAGE.

Cusick, J. J. (2019). *A Survey of Maturity Models from Nolon to DevOps and Their Applications in Process Improvement.*

Dam, K. v., Oreg, S., & Schyns, B. (2008). Daily work contexts and resistance to organisational change: The role of leader–member exchange, development climate, and change process characteristics. *Applied Psychology, 2*(57), 313-334.

Diaz, J., Almaraz, R., Pérez, J., & Garbajosa, J. (2018). DevOps in practice: an explanatory case study. *Proceedings of the 19th International Conference on Agile Software Development: Companion (XP '18)*.

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

Ebert, C., Gallardo, G., Hernantes, J., & Serrano, N. (2016). DevOps. *Ieee Software, 33*(3), 94-100.

Edwards, D. (2010). What is DevOps?

Elliot, S. (2014). DevOps and the cost of downtime: Fortune 1000 best practice metrics quantified. *International Data Corporation (IDC)*.

Feijter, R., Overbeek, S., Vilet, R., Jagroep, E., & Brinkkemper, S. (2018). DevOps competences and maturity for software producing organizations. *Enterprise, Business-Process and Information Systems Modeling*, 244-259.

Feijter, R., Vliet, R., Jagroep, E., Overbeek, S., & Brinkkemper, S. (2017). Towards the adoption of DevOps in software product organziations: A Maturity model approach. *Technical Report Series, (UU-CS-2017-009)*.

Feilzer, Y. M. (2010). Doing Mixed Methods Research Pragmatically: Implications for the Rediscovery of Pragmatism as a Research Paradigm. *Journal of Mixed Methods Research, 4*(1), 6-16.

Flick, U. (2004). Triangulation in qualitative research. *A companion to qualitative research, 3*, 178-183.

Gasparaité, M., & Ragaišis, s. (2019). Comparison of devops maturity models. *IVUS*.

Ghantous, G. B., & Gill, A. (2017). DevOps: Concepts, Practices, Tools, Benefits, and Challenges. *PACIS 2017 proceedings*.

Gill, A. Q., Loumish, A., Riyat, I., & Han, S. (2018). DevOps for information management systems. *VINE Journal of information and knowledge management systems*.

Goldkuhl, G. (2012). Pragmatism vs interpretivism in qualitative information systems research. *European Journal of Information Systems, 21*(2), 135-146.

Holland, J., Thomson, R., & Henderson, S. (2006). *Qualitative Longitudinal Research: A discussion paper*. London: London South Bank University.

Inbar, S., Yaniv, S., Gil, P., Eran, S., Olga, K., & Ravi, S. (2013). DevOps and OpsDev: How Maturity Model Works.

Jalali, S., & Wohlin, C. (2012). Systemativ Literature Studies: database searches vs. backward snowballing. *Proceedings of the 2012 ACM-IEEE International Symposium on emperical software engineering and measurement*, 29-38.

König, L., & Steffens, A. (2018). Towards a Quality Model for DevOps. *Continuous Software Engineering & Full-scale Software Engineering*, 37-43.

Kersten, M. (2018). A cambrian explosion of DevOps tools. *IEEE Annals of the History of Computing*(2), 14-17.

Khoshgoftar, M., & Osman, O. (2009). Comparison of maturity models. *2009 2nd IEEE International Conference on Computer Science and Information Technology*, 297-201.

Kim, G. (2018). Top 11 things you need to know about DevOps.

Krancher, O., Luther, P., & Jost, M. (2018). Key Affordences on Platform-as-a-Service: Self-Organzation and Continuous Feedback. *Journal of Management Information Systems, 35*(3), 776-812.

Kruuse, E. (2007). *Kvalitative forskningsmetoder I psykologi og beslægtede fag.* København: Akademisk forlag.

Lan, P. (2018). A review of key paradigms: positivism, interpretivism and critical inquiry.

Langley, A. (1999). Strategies for theorizing from process data. *Academy of Management review, 4*(24), 691-710.

Langley, A., & Truax, J. (1994, September). A process study of new technology adoption in smaller manufacturing firms. *Journal of management studies, 5*(31).

Lee, G., & Xia, W. (2006). Organizational size and IT innovation adoption: A meta-analysis. *Information & Management, 8*(43), 975-985.

Lu, Y., & Ramamurthy, K. (2011). Understanding the link between information technology capability and organizational agility: an empirical examination. *MIS Quarterly, 35*(4), 931-954.

Lwakatare, L. E., Kuvaja, P., & Oivo, M. (2015, May). Dimensions of DevOps. *International Conference on Agile Software Development*, 212-217.

Mamatha, C., & Kiran, S. R. (2018). Implementation of DevOps Architecture in the project development and deployment with help of tools.

Mohamed, S. I. (2015). DevOps shifting software engineering strategy: Value based perspective. *IOSR Journal of Computer Engineering (IOSR-JCE), 17*(2), 51-57.

Mohamed, S. I. (2016). DevOps maturity calculater DOMC-value oriented approach. *International Journal of Engineering Research & Science (IJOER), 2*(2).

Nielsen, P. A., Winkler, T. J., & Nørbjerg, J. (2017). Closing the IT Development-Operations Gap: The DevOps Knowledge Sharing Framework. *BIR Workshops*.

Nowell, L. (2015). Pragmatism and integrated knowledge translation: exploring the compatabilities and tensions. *Nursing Open, 2*(3), 141-148.

Pfeffer, J., & Sutton, R. I. (2006). Evidence-based management. *Harvard Business Review, 84*(1), 62.

ProActive. (2020). *IntraActive*. Retrieved from IntraActive: https://intraactive.dk/

ProActive. (2020). *Om Os*. Retrieved from ProActive: https://www.proactive.dk/om-os

Ramanujan, S., & Kesh, S. (2004). Comparison of knowledge management and CMM/CMMI implementation. *Journal of American Academy of Business, 4*(1/2), 271-275.

Reed, P. J. (2015). *DevOps in Practice.* O'Reilly Media.

Riungu-Kalliosaari, L., Mäskinen, S., Lwakatare, L. E., Tiihonen, J., & Männistö, T. (2016, November). DevOps adoption benefits and challenges in practice: a case study. *Internation Conference on Product-Focused Software Process Improvement*, 590-597.

Rong, G., Zhang, H., & Shao, D. (2016, May). CMMI guided process improvement for DevOps projects: an exploratory case study. *Proceedings of the International Conference on Software and Systems Process*, 76-85.

Royce, W. (2002). CMM vs. CMMI: From conventional to modern software management. *The Rational Edge, 2*.

Shibab, E., Bird, C., & Zimmermann, T. (2012, September). The effect of branching strategies on software quality. *Proceedings of the ACM-IEEE International Symposium on Empirical software engineering and measurement*, 301-310.

Silva, C. C., Gilson, F., & Galster, M. (2019, November). Comparison Framework for Team-Based Communication Channels. *International Conference on Product-Focused Software Process Improvement*, 315-322.

Skelton, M., & Pais, M. (2019). *Team Topologies: Organizing Business and Technology Teams for Fast Flow.* IT Revolution.

Smeds, J., Nybom, K., & Porres, I. (2015, May). DevOps: a definition and perceived adoption impediments. *International Conference on Agile Software Development*, 166-177.

Smith, D. M. (2011). *Hype cycle for cloud computing.* Gartner Inc., Stamford.

Tallon, P. P., Queiroz, M., Coltman, T., & Sharma, R. (2019). Information technology and the search for organizational agility: A systematic review with future research possibilities. *The Journal of Strategic Information Systems, 2*(28), 218-237.

Virmani, M. (2015). Understanding DevOps & bridging the gap from continuous integration to continuous delivery. *Fifth International Conference on Innovative Computing Technology (INTECH)*, 78-82.

Walls, M. (2013). *Building a DevOps culture.* O'Reilly Media, Inc.

Webster, J., & Watson, R. T. (2002). Analyzing the past to prepare for the future: Writing a literature review. *MIS Quaterly*, 13-23.

Whyte, J., Stasis, A., & Lindkvist, C. (2016). Managing change in the delivery of complex projects: Configuration management, asset information and 'big data'. *International Journal of Project Management, 2*(34), 339-351.

Wiedemann, A., Forsgren, N., Wiesche, M., Gewald, H., & Krcmar, H. (2019). Research for practice: the DevOps phenomenon. *Communications of the ACM, 62*(8), 44-49.

Winkler, T. J., & Günther, O. (2012). Explaning the Governance of Software as a Service Applications: A Process View. *Multikonferenz der Wirtschaftsinformatik (MKWI) 2012 Proceedings*, 599-612.

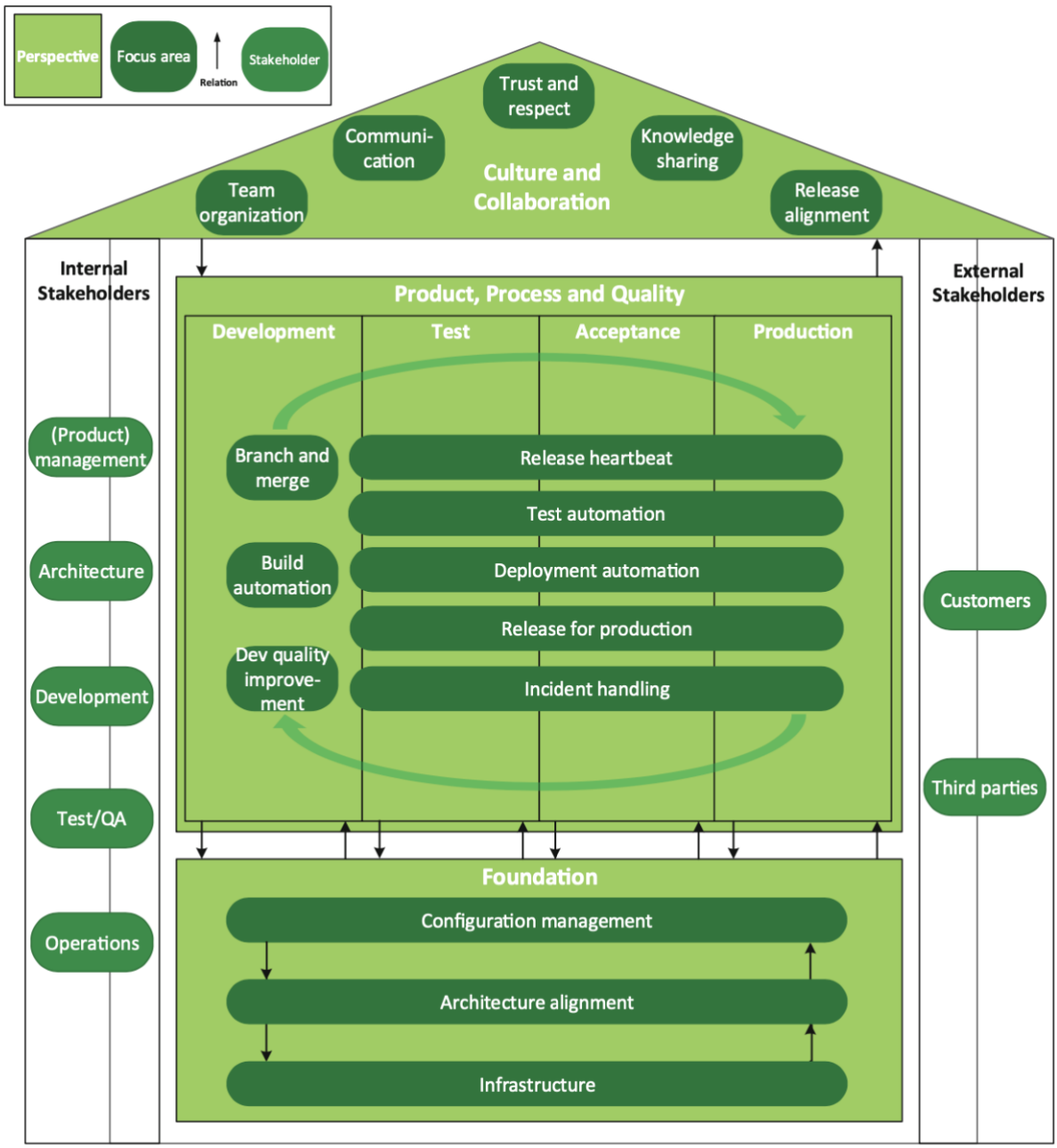
Yin, R. (2002). *Case Study Research – Design and Methods.* London: SAGE Publications.

Zarour, M., Alhammad, N., Alenzi, M., & Alsarayrah, K. A. (2019). Research on DevOps Maturity Models.

Appendices

A

11.1. Appendix A – Competence Model (Feijter et al. 2018)



11.2. Appendix B – Focus Areas and Capabilities (Feijter et al., 2018)

CC - Communication

A. Indirect communication communication between interdisciplinary professionals, among which are Dev and Ops professionals, is indirectly established (e.g. through managers, procedures). **B. Facilitated communication** direct communication between interdisciplinary professionals, among which are Dev and Ops professionals, is facilitated by management by stimulating professionals to communicate directly. **C. Direct communication** direct interdisciplinary communication between professionals, among which are Dev and Ops professionals while working towards a release is present. This direct communication could occur through mailing lists, per- sonal contact etc. **D. Structured communication** a structure for interdisciplinary communication is in place (e.g. by holding daily standups and retrospectives with interdisciplinary professionals including Dev and Ops, and by maintaining contact with (product) management to discuss about impediments along the way, work to be done the upcoming sprints, and the technical debt situation, among others). **E. Communication improvement** communication among management and interdisciplinary professionals, including Dev and Ops, is improved (e.g. by adopting and trying out new communication practices from industry, learning from experiences and by tracking projects or using instruments such as skill matrices and peer feedback mechanisms over time).

CC - Knowledge sharing

A. Decentralized knowledge sharing knowledge is shared between interdisciplinary professionals, among which are Dev and Ops professionals in a decentralized way (i.e. through notes or documents). **B. Centralized knowledge sharing** knowledge is shared between inter- disciplinary professionals, among which are Dev and Ops professionals, through centralized knowledge sharing fa- cilities. **C. Active knowledge sharing** knowledge is shared actively between interdisciplinary professionals, among which are Dev and Ops professionals. **D. Communities of practice** knowledge is shared through communities of practice, which are composed of multidisciplinary professionals that share a common interest.

CC - Trust and respect

A. Culture of trust and respect imitation dynamics, level of autonomy, and planning are open for collaboration and creation of trust and respect between interdisciplinary professionals, among which are Dev and Ops people. An example here is a DevOps duty rotation where developers take on operational tasks. **B. Culture of trust and respect facilitation** a culture of trust and respect is facilitated by management. Facilitation by management means that management should not manage by fear, but should act as a servant leader that supports professionals in day-to-day tasks, has an understanding of operational tasks, and allows interdisciplinary professionals, among which are Dev and Ops professionals, to learn quickly from mistakes. **C. Culture of trust and respect shared core values** the culture of trust and respect between interdisciplinary professionals, among which are Dev and Ops professionals, is maintained by following shared core values such as rewarding Dev and Ops as a group when a release is successful, being transparent and open towards one another to prevent blaming, and working towards shared goals.

CC - Team organization

A. Separate teams separate teams are present (e.g. development teams, operations teams etc). **B. Cross functional teams excluding Ops** cross functional teams are present that exclude operations (e.g. teams consisting of developers and testers are present). **C. Cross functional teams including Ops** cross functional teams are present that include operations. **D. Cross functional teams with knowledge overlap** cross functional teams are present in which professionals have boundary crossing knowledge (e.g. T-shaped professionals that have Dev and Ops knowledge).

CC - Release alignment

A. Roadmap alignment alignment with dependent internal and external stakeholders (e.g. third parties) is considered in the roadmap. **B. Internal release heartbeat alignment** the release heartbeat is aligned with dependent internal stakeholders. An example of such an alignment could be reflected in adopting the same deployment moments or adhering to a common sprint cadence. **C. External release heartbeat alignment** the release heartbeat is aligned with dependent external stakeholders such as third parties from which software is used in the development of a product.

PPQ - Release heartbeat

A. Requirements and incidents gathering and prioritization Functional and nonfunctional requirements and incidents are gathered from and prioritized with internal stakeholders and external stakeholders (e.g. customers). **B. Fixed release heartbeat and validation** a fixed release heartbeat is present and validation of functionality occurs with internal stakeholders and external stakeholders (e.g. customers) by demoing the functionality on a test or acceptance environment or the like. **C. Production requirements and incident gathering** functional and nonfunctional requirements and incidents are gathered from production by monitoring the production environment(s). **D. Gradual release and production validation** functionality is released gradually (e.g. functionality is first released to internal stakeholders, whereafter it is released to stakeholders that have close bonds with the organization. Finally, the software is released to end-customers) and validation of functionality occurs in production. **E. Feature experiments experiments** are run with slices of features in order to support the prioritization of the contents in the backlog (e.g. A/B testing). **F. Release heartbeat improvement** the value stream is continuously improved by identifying and eliminating activities that do not add any value, shortening lead times and shortening feedback loops such as the time between feedback moments with the customer.

PPQ - Branch and merge

A. Version controlled source code source code is stored under version control. **B. Branching/merging strategy** a branching/merging strategy is adhered to that allows multiple developers to collaborate and allows code to be branched and merged. **C. DevOps branching/merging strategy** a branching/merging strategy is adhered to that is DevOps compatible. An example of such a strategy is trunk based development. **D. Feature toggles** feature toggles are used to release functionality to customers by making completed functionality available.

PPQ - Build automation

A. Manual build creation a software build is created manually. **B. Automated build creation** a build is cre- ated automatically (e.g. by running a scheduled build at night). **C. Continuous build creation** a CI build is created after each check-in to verify that the integrated code still yields a working software build.

PPQ - Development quality improvement

A. Manual code quality monitoring manual code quality improvement mechanisms are in place such as pair programming, code reviews, and adherence to code conventions. **B. Broken build detection** broken software builds are detected, made visual and quickly repaired. **C. Gated check-in** gated check-ins are performed. **D. Automated code quality monitoring** code quality is monitored automatically (e.g. automated code reviews**). E. Quality gates** quality gates are defined against which the quality of code is measured.

PPQ - Test automation

A. Systematic testing Manual unit and acceptance tests are performed systematically. **B. Advanced systematic testing** manual integration (chain) and regression tests are performed systematically and test driven development practices are used in testing such as using mocking frameworks and writing unit tests before writing code. **C. Automated systematic testing** automated unit and nonfunctional tests are performed systematically. **D. Advanced automated systematic testing** automated regression, integration (chain) and acceptance tests are performed systematically. **E. Automated recoverability and resilience testing** automated recoverability and resilience tests are randomly performed in production.

PPQ - Deployment automation

A. Manual deployment software is deployed to environments in a manual fashion. In addition, rollback is possible, where data is brought back to a stable state. **B. Partly automated deployment** software is deployed automatically to some environments. **C. Continuous delivery** deployment to all environments occurs in an automated manner (e.g. via self service deployments), where data model changes are also processed automatically. **D. Continuous deployment** each check-in is continuously deployed to production, where data model changes are also processed and automated rollback is possible.

PPQ - Release for production

A. Definition of done a definition of done that incorporates development and testing criteria, among others to be complied with during a sprint, is followed. **B. Definition of release** a definition of release that incor- porates Ops criteria (e.g. verifying whether the software works in production) to be complied with before releasing to customers, is followed. **C. Done according to customer**

functionality is declared done when customer satisfaction has been reached. **D. Automated material generation** Supporting materials such as release documentation, training documentation etc. are automatically generated.

PPQ - Incident handling

A. Reactive incident handling incidents are reactively acted upon by interdisciplinary professionals, among which are Dev and Ops professionals. **B. Proactive incident handling** incidents are proactively acted upon by interdisciplinary professionals, among which are Dev and Ops professionals **C. Blameless root cause detection** root causes are identified without blaming one another by conducting blameless postmortems involving both Dev and Ops. **D. Automated root cause detection** the identification of root causes of incidents is supported by analytics.

F - Configuration management

A. Manual configuration management Supported versions of configuration items (e.g. OS, middleware etc.) and their relationships are managed manually, for instance in documents or excel sheets. **B. Automated configuration management** Supported versions of con- figuration times and their relationships are managed in a configuration management tool. **C. Version controlled configuration management** Supported versions of the configuration items and their relationships are managed in version control.

F - Architecture alignment

A. Software and technical architecture alignment the software architecture of an application is aligned with a technical architecture before a release. **B. Continuous architecture evolvement** the software and technical architecture evolve mutually in a continuous fashion in such a way that these architectures are continuously aligned and kept up to date.

F - Infrastructure

A. Manually provisioned infrastructure infrastructure such as development, test, acceptance and produc- tion infrastructure is available and provisioned manually. **B. Partly automatically provisioned infrastructure** A part of the infrastructure between development and production is

equivalent in terms of configuration and hardware and some or all environments are provisioned automatically. **C. Automatically provisioned infrastructure** infrastructure between development and pro- duction is equivalent in terms of configuration and hard- ware and provisioned automatically. **D. Managed platform services platform services** (such as a web server and a database server) are preconfigured in the plat- form and allow for applications being directly deployed, among others, while rights and rolls are managed per en- vironment. This is also known as platform as a service.

11.3. Appendix C – Interview Guide

Area of Concern	Question
Introduction	• What is your current job description? • How long have you been at this company? • Can you describe your tasks and responsibilities?
The Company	• How would you describe the company culture? • What is the organizational IT structure?

The Teams	• Where are the different teams located?
	• How do you communicate within the team?
	  ○ Do you use any special methods or communication channels (mail, chat, etc.)?
	  ○ Do you use any specific knowledge sharing platform?
	• When the team was established did you notice any differences in culture between the teams that were put together?
	  ○ Was there done any effort in bringing down the formal wall between the entities?
	  ○ Do you feel that all team members have respect for and listen to one and other?
	• Is there a sense of a common goal within the team?
	  ○ Are the teams rewarded in any specific way? What are they measured on?

The Process	• What is the process from a feature in development to production? • How do you decide what features to develop for customers? • How often do you deploy new features? • Is every team member included in the entire process from development to deployment? • How long time does it take for a finished feature to reach the customer?

	• How much of your process from development to production is automated (integration, test, deployment)? • How do you monitor your software/product? ○ If something goes wrong in production how do you handle it? • Do you experience less errors after adopting DevOps?
Challenges	• Have you encountered any large problems within the team? • Is that something that hinders you from progressing with DevOps? • Do you feel like your team size or organizational structure has had any effect on your approach to DevOps?
Value	• Has your time to market been reduced after the introduction to DevOps? • Do you see it as an advantage that you can deploy more often than before? • Do you feel that the introduction of DevOps has improved your organizational branding (more technological/innovative brand)? • Have your customers been more involved after the introduction of DevOps? • Has your customer satisfaction increased? • Are customers involved in the process of developing features? • Do you feel that the transition to DevOps has provided any value? And is there more to gain?
Final Questions	• Is there anything you want to add? • What do you think is the most important thing when discussing DevOps?

11.4. Appendix D – Primary Data

Please refer to the uploaded .zip-file for access to the primary data

List of Respondents

Interview transcripts 2020: ProActive A/S 1

Respondent 1 – Developer 2

Respondent 2 – Senior Developer 8

Respondent 3 - Tester 14

Respondent 4 – Technical Product Delivery Manager 17

Respondent 5 – Technical Product Delivery Manager (Retrospective interview) 26

Interview transcripts 2020: Topdanmark A/S 29

Respondent 6 - Developer 29

Respondent 7 – Developer 33

Respondent 8 – Product Owner 39

Respondent 9 – Hawks (Developer) 47

Respondent 10 – IT Development Manager and Architect 53

Respondent 11 – Scrum Master 61

Respondent 12 – DevOps Specialist in IT Operations 66

Respondent 13 – IT Development Manager and Architect (Retrospective Interview) 71

Document: Primary Data

11.5. Appendix E – Secondary Data

Please refer to the uploaded .zip-file for access to the secondary data (Nielsen et al., 2017)

List of Respondents

Interview transcripts 2015: ProActive A/S 1

Respondent 14 – Director Solutions (IT Development) 2

Respondent 15 – Product Owner (IT Development) 11

Respondent 16 – Developer/Solutions Architect (IT Development) 26

Respondent 17 – Tester (IT Development) 41

Respondent 18 – IT Professional (IT Operations) 58

Respondent 19 – Director Solutions (IT Development / Follow-up Interview) 74

Interview transcripts 2015: Topdanmark A/S 80

Respondent 20 - Arkitektur & Metode/Agil Udvikling & Test – Service Owner (SE) 80

Respondent 21 – Arkitektur & Metode /Agil Udvikling & Test – Specialist 98

Respondent 22 - IT Skade (IT Development) – Product Owner (PO) 112

Respondent 23 - IT Skade (IT Development) – Application and Architecture Responsible (AAA) 129

Respondent 24 – IT Skade (IT Development) – Developer 142

Respondent 25 – Applikationsplatform / IT Process Management (IT Operations) – Service Owner (SE), Specialist, Release Manager (RM) 163

Respondent 26 - IT Metode (IT Operations) –Service Owner (SE) 191

Document: Secondary Data