INSTITUTO
UNIVERSITÁRIO
DE LISBOA

DevOps Applied in Change Management Process

Victor Afonso Camargo

Master in Computer Science and Business Management

Supervisor:
PhD. Rúben Filipe de Sousa Pereira, Auxiliar Professor
ISCTE - Instituto Universitário de Lisboa

Co-supervisor:
João Pedro Carvalho Faustino,
Deloitte Portugal

November, 2022

TECHNOLOGY
AND ARCHITECTURE

Department of Information Science and Technology

DevOps Applied in Change Management Process

Victor Afonso Camargo

Master in Computer Science and Business Management

Supervisor:
PhD. Rúben Filipe de Sousa Pereira, Auxiliar Professor
ISCTE - Instituto Universitário de Lisboa

Co-supervisor:
João Pedro Carvalho Faustino,
Deloitte Portugal

November, 2022

Acknowledgements

This study and all the methodology it entailed would not be possible without the support of my supervisor, Prof. Dr. Rúben Pereira. His availability and patience to review every version of this document from the earliest stages to its final result was significant to . Also, for making sure I was always on track and reached the deadlines.

To my co-supervisor João Faustino, whose feedback was pivotal to design this study. Thank you for the given feedback at every stage and availability to discuss the many aspects of this research.

To my family and friends for encouraging me to keep going and accomplish this endeavor. Special thanks to Vivian Kaiser, Marco Moreira and José Dantas for the encouragement during the entire process. Thank you for listening. Shaun, my love, thank you for the emotional support and patience during these challenging times.

To my incredible team at work and each contributor of this research, I am deeply thankful. This would never be possible without you. I am more than grateful that you provided your precious time and make this study a reality. Finally, to my company which supported me, by providing me the means I needed to achieve this goal.

Thank you for being part of this phase in my life.

Resumo

Vivemos em um ambiente em rápida mudança e para obter uma resposta mais rápida ao mercado com maior confiabilidade, onde as equipas de desenvolvimento de software precisam organizarem-se, pois tem o desafio de entregar mudanças com qualidade. As metodologias ágeis surgiram como uma resposta para lidar com esse ambiente dinâmico, mas com foco no desenvolvimento de software, deixando para trás sua componente operacional. DevOps surgiu como um conjunto de práticas e valores culturais para preencher a lacuna existente entre desenvolvimento e operações.

Contudo, para garantir conformidade e governança adequadas, as organizações precisam gerir as mudanças na velocidade do negócio e, ao mesmo tempo, atender aos critérios de conformidade. O Change Management aborda esse tópico estabelecendo processos para gerir essas mudanças.

O objetivo deste estudo é investigar como as práticas de DevOps podem relacionar-se com o processo de Change Management. Portanto, foi realizado um estudo de caso em uma empresa orientada por práticas de DevOps para explorar o potencial relacionamento sobre esses temas. As principais descobertas deste estudo demonstram que, ao entregar mudanças por meio de processos automatizados, a qualidade deve ser garantida. As etapas do Change Management devem ser seguidas baseadas na criticidade da mudança. As práticas que estão em sua maioria dentro do âmbito da automação na entrega de software sugeriram estar mais alinhadas ao processo de Change Management. Além disso, as práticas que exigem colaboração entre os membros da equipa desempenham um papel importante, principalmente ao comunicar a implementar de tais mudanças.

Palavras-Chave: DevOps; Change Management.

Abstract

We live in a rapid-changing environment; and to achieve a faster response to the market with higher reliability, software development teams must be able to organize themselves. The challenge to implement these changes in production is to ensure quality in its deliverable. Agile methodologies emerged as a response to address this dynamic environment. However, these methodologies focused mainly on the software development side, leaving its operations counterpart behind. DevOps emerged as a set of practices and cultural values aiming to bridge the existing gap between development and operations.

Still, in order to ensure compliance and proper governance, organizations need to manage changes at the speed of business while meeting their compliance criteria. Change management addresses this topic by establishing processes to create and deploy these changes.

The aim of this research is to investigate how DevOps practices can relate to the change management process. This way, a case study at a company driven by DevOps practices was conducted to explore the potential relationship about these topics. The main findings of this research demonstrate that in delivering changes through automated processes, quality must ensured. The change management steps should be followed based on how critical the change is. Practices that are mostly within the scope of the automation in software delivery process suggested to be more aligned to the change management process. Also, practices that require collaboration among the team members play an important role, especially when communicating the deployment of such changes.

Keywords: DevOps; Change Management.

Contents

List of Tables

List of Figures

Abbreviations

CAB Change Advisory Board

CD Continuous Delivery

CI Continuous Integration

CM Change Management

CS Case Study

IT Information Technology

ITIL Information Technology Infrastructure Library

RFC Request for Change

RQ Research Question

SDLC Software Development Life-cycle

CHAPTER 1

Introduction

We live in a fast-paced digital landscape where changes are inevitable [1]. To reach time-to-market and be able to deploy changes to production faster, organizations must have efficient methodologies and processes incorporated in their Software Development Life Cycle (SDLC) [2].

In software development, there are key problems, such as long development cycles, increasing costs and lack of quality upon delivery, which agile methods aim to solve [3]. Methodologies such as Extreme Programming (XP) [4] and SCRUM [5] introduced a way to react to changes, leading to improvements in development time, communication, collaboration, especially in rapidly-changing organizations where reaching deadlines is critical to create market advantage [6], [7], [8].

Agile systems are characterized by their ability to change cost-effectively and quickly [9]. These software development techniques have gained broad acceptance and have significantly contributed to how development projects are conducted, especially in dynamic environments [10].

In the mindset of an agile team, changes from customers are a welcome part of the development process. This ensures flexibility in design and development [11]. Resources are effectively and efficiently assigned to a specific task, leading to overall customer satisfaction. [12].

Over the years, this agile transformation has been adopted by many organizations, albeit primarily with a focus on the software development process, leaving the operations side in the optimization race behind [13]. As a result, the operations side was left out of this transformation [14].

DevOps emerged as a response to the growing understanding that there is a gap between development and operations, which Houssini et. al expanded in the 4Cs (communication, cooperation, culture, and collaboration) between an IT development function and an IT operations function, in which both teams need to work harmoniously [15].

Therefore, DevOps introduces a set of practices that motivate businesses to make a transition to this new methodology [16]. Not only does it presents a set of practices, but also capabilities

such as skills and knowledge [17], that evolve dynamically over time, where team collaboration and communication are considered the most critical components [18].

However, changes emerge during the SDLC and it is critical that organizations guarantee that compliance is met alongside proper governance while managing changes at the speed of business [19]. This concern can be addressed by change management, which can be defined simply as "*(the) processes to ensure strategic changing in organization systematically for handling resistance a change and archive business goal for transformation*" [20].

Traditional implementations introduce massive layers of bureaucracy and impediments, counteracting DevOps principles and slowing down business velocity [1]. The scope of changes in software development are extensive, and their occurrences may be direct or indirect, e.g. network changes, operation system upgrades, and other unknown changes that are part of software development [21].

Therefore, one can imply that if a change management process does not evolve, the organization may be left behind. Moreover, based on what was stated previously, this may be contradictory, leading us to wonder: *how can we manage changes in such fast-paced environments?*, and even further; *what is the relationship between DevOps and change management?*. That said, this investigation aims to understand the link between DevOps and change management processes supported by a case study and unveils the answers to these questions.

This research is structured as follows; the *Theoretical Background* chapter describes the fundamental concepts that frame this study and attempts to introduce the readers to the concepts previously explored; the following chapter is *Literature Review*, in which the author examines the existing literature and the related work revolving around the objectives mentioned earlier and addresses the need of this study; then, *Research Methodology* defines the protocols of the subsequent case study; next, *Case Study Analysis and Reporting*, demonstrates the methods the author used to collect the data for a further analysis; finally, the author presents the *Conclusions* based on the results during the analysis stage and also reiterates the relevance of this research on the academic and professional areas.

CHAPTER 2

Theoretical Background

This chapter presents the fundamental topics in this study: Agile Software Development, DevOps and Change Management (CM).

2.1. Agile Software Development

Agile software development unfolded from the personal experiences and collective wisdom of consultants and practitioners in the software community, which resulted in a set of principles expressed in the Agile Manifesto [1]. It motivates and empowers software engineers to create business value by delivering working software to users at regular, small intervals [22].

Dingsøyr et al. also argue that these principles are not a formal definition of agility, but rather guidelines for delivering high-quality software in an agile manner. Also, it encourages practices that accommodate change in requirements at any stage of the development process [22]. The concept of agility and its components are subject to definition from many practitioners and authors. For instance, [23] and [24] cite the ability to rapidly and flexibly create and respond to change in the business and technical domains. More formally [23] et al. later state it is "*the ability to adapt to different changes and to refine and fine-tune development processes as needed*". The definition considered for this research is "*a reaction to plan-based or traditional methods*, which emphasizes "a rationalized, engineering-based approach" incorporating extensive planning, codified processes, and rigorous reuse [25]. By contrast, agile methods address the challenge of an unpredictable world by recognizing the value competent people and their relationships bring to software development [26] [27].

Since the manifesto and principles for Agile Software Development were published in 2001, popular approaches have been derived, Extreme Programming (XP) and Scrum. These approaches are still successfully employed today and are considered standard development methodologies [28].

[1] See `https://agilemanifesto.org/`, accessed on Nov 30th

The core practices of Agile methodologies are as follows: small, short releases, stakeholders physically located together, and a time-boxed project cycle [29], which contributes to the success of the projects [30]; [28])

2.2. DevOps

Nowadays, IT Organizations are dealing with significant challenges, such as enabling business to respond in the fastest way possible to market conditions and customer expectations, as well as working and maintaining also working and maintaining complex systems that need to be highly-available and reliable [31]. To address the constraints of bringing new functionalities to market faster and delivering highly-available systems, organizations had to find a new way to integrate development and operations [31].

This study considers the DevOps definition introduced by Erbert as *"[DevOps] is about fast, flexible development and provisioning business processes. It efficiently integrates development, delivery, and operations, thus facilitating a lean, fluid connection of these traditionally separated silos."* [32].

Despite the definition above and its similarity to Agile, a formal definition for DevOps is not common, however [31] noted four principles:

- Culture — it relies on a cultural shift by accepting joint responsibility, meaning it fosters a cultural shift toward collaboration between development, quality assurance and operations.
- Automation — it relies on full automation of the build, deployment and testing in order to achieve short lead times, and consequently rapid delivery and feedback from end users.
- Measurement — through measuring, it aims to improve the processes regarding delivery capabilities based on the project goals.
- Sharing — it develops different aspects, for instance knowledge, tools and infrastructure, while also empowering the team by celebrating the successful releases bringing them together.

2.3. Change Management

IT service management (ITSM) aims to achieve quality customer service by ensuring that customer requirements and expectations are aligned throughout the development phase [33]. CM is a crucial part in the ITSM implementation as it requires working collaboratively across many departments

within an organization [34]. Despite this, Wui-Gee Tan et al. also note its challenges such as resistance and an atmosphere of negativity and skepticism when CM issues arise [34]. This research adopts the CM definition so that "*(it) ensures that standardized methods and procedures are used for efficient and prompt handling of all changes, in order to minimize the impact of any related incidents. To achieve this requires a careful and considered approach to assessing risk, potential impact of change, the resource requirements, and the process for approving changes.*" [35].

Regarding the changes, they must be tracked, reported, assessed and implemented within the scope defined by the change approval board (CAB) [36]. The three types of changes [37] in information systems are described below:

- Standard Change — these are low-risk, pre-authorized changes that are well understood and fully documented, and can be implemented without needing additional authorization.
- Normal Change — the changes do need to be scheduled, assessed and authorized following a process.
- Emergency change — these are changes that must be implemented as soon as possible, for example an incident or a security patch.

CHAPTER 3

Literature Review

Given that this research intends to study DevOps as applied in the CM process, it is crucial to conduct research on the existing literature. However, as DevOps is a recent phenomenon [16] and we intend to associate it to CM, the author decided to perform a literature review. This chapter introduces the literature review performed and its methodology, followed by the findings discussed in the upcoming sections.

This chapter is organized as follows, in Section 3.1 the author defines the methodology used in this literature review and why it is required. It is followed by Section 3.2 where the review planning is detailed. In Section 3.3 the author outlines how the review was conducted. Finally, in Section 3.4, the findings are reported and synthesizes the related work. This final section also presents a Table 2 detailing the concepts found in the reviewed literature, followed by a discussion on the key findings of this review.

3.1. Methodology

A multi-vocal literature research (MLR) approach is needed where evidence on a given topic in the academic domain is lacking [38]. It consists of a type of Systematic Literature Review (SLR), that includes Grey Literature (GL) and can be very useful as it provides an overview of the state of the art and practice in a target area [39]. Garoussi et al also point out two its benefits in software engineering: (1) significant advantages in some software engineering areas and (2) evidence of the challenges faced by practitioners [39]. The research protocol for this study is shown in Figure 1.

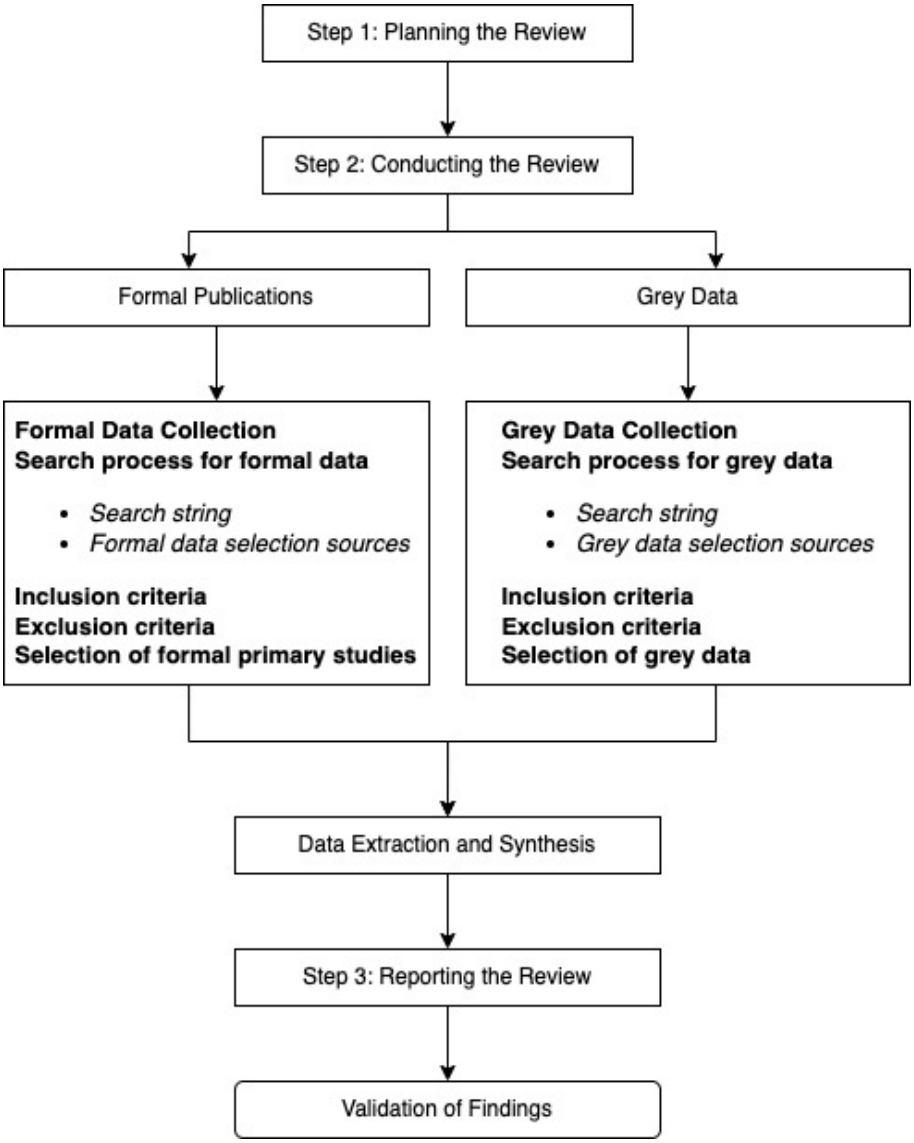

Figure 1. *MLR search protocol*

3.2. Planning the Review

In this section, we present the need for this research as well as outline the goal and the evaluation of this multi-vocal literature review. The main focus of this literature review is to understand what other researchers and practitioners have been doing in regards to the relationship between DevOps and the CM process.

At this point we will define a search string and the libraries to search for documents. This way, we will dive into a vast amount of research that addresses the research goal, defined in the last chapter. The search string and libraries used in this study are listed below.

Search String:	"DevOps" AND "Change Management"
Libraries:	EBSCO; Elsevier Scopus; IEEE Xplore; Web of Science
Grey Literature Sources:	Google Search Engine [2]

The a filtering criteria was applied to each document as presented on Table 1

Table 1. *Document Filtering Criteria*

Filter	Description
F_0	All documents matching the keywords.
F_1	Filter documents since 2014.
F_2	Filter documents only written in English.
F_3	Filter documents.
F_4	Manual filtering.

3.3. Conducting the Review

To conduct the review, the author utilized a two-fold approach as Figure 1 illustrates. The author searched for formal publications on the following research databases: Elsevier Scopus, Web of Science, EBSCO Host and IEEE Explorer, and then on a search engine. This search was also conducted in tandem with another search for grey literature, repeating the search criteria outlined in the previous section. The keywords 'DevOps' and 'Change Management' were employed in this study. Considering the inclusion of the grey literature, the author decided to use the Google search engine [3] using the same keywords mentioned earlier. Finally, the time frame of this research was between October 2021 and January 2022.

[3]See `https://www.google.pt/`, accessed on Nov 30th

The search returned 27 documents, excluding duplicated articles. Only the relevant works for this research were considered. The excluded documents, even those with having matching keywords, were removed for the purpose of this literature review due to their absence of significant content related to our review.

3.4. Reporting the Review

In this section, the author describes the findings in the literature and summarizes the related work. The main goal is to identify the topics found in the articles associated with their respective authors. The author of this study defined a set of prevalent concepts in the reviewed content, in order to facilitate the understanding between the connections of the articles. The information found will be discussed in the following sections supplemented by the concept matrix represented by Table 2.

3.4.1. Automation, Continuous Integration and Continuous Delivery

As we can observe in Table 2, most articles refer to automation as a considerable aspect when DevOps is applied in a CM process. For instance, automation is essential in DevOps CM [40], and its role facilitates continuous integration and development process [41].

Another important concept that is related to automation is Continuous Integration (CI) and Continuous Delivery (CD). In this case, changeCM is the catalyst to progress in the CI/CD pipeline that will deploy the changes in the code [42]. Despite the benefits of having a working pipeline to deliver and deploy software, it should be used judiciously. Furthermore, "continuous" does not necessarily equate to "all the time" or even "as fast as possible"; rather, it means being as fast and practical as possible given the organization's business needs for the specific situation [43]. In addition to these advantages, it is also possible to ensure safer and faster deployments by having canary testing and a staged deployment initiative [21]. This is also supported by [44], as it states that if something is amiss and the impact is manageable, the correction can be performed with minimal overhead.

3.4.2. Metrics and Monitoring

Frequent code changes demand visibility, so continuous monitoring and metrics is another concept to be included in this review. It allows teams to react swiftly to business opportunities [45] and unveils issues before they become visible to the end user [46]. It should be as automated as possible, triggering alerts automatically when needed [21]. Employing continuous monitoring and alerting

creates an opportunity to assess the maturity of the organization [41] and by doing so, it establishes a CM plan in the realm of incident management [45]. Another study found in the literature to address this topic was Harmonia, a continuous service monitoring framework to improve coordination and CM among independent teams who worked with a micro-services architecture [47].

3.4.3. Cultural Shift and Communication

Culture shift was a noteworthy aspect found in many studies, and is suitable when the combination of DevOps and CM are not at odds with each other, but rather an ideal cultural match [48]. In [45], there is a acknowledgement of the change in the enterprise delivery and processes mindset. This way, with the assistance of automation, change and release management becomes an integrated process [49]. Communication itself is another concept explored in the literature, and having unified communication is an effective way to foster collaboration [50]. However, it comes with a challenge [51]: when it approaches the CM sphere, it may lead to tension between the CAB and the developers [19].

3.4.4. Task Prioritization and Strategic CAB

From a business strategy perspective, the author of this study identified two elements: task prioritization and a strategic CAB. A strategic CAB should be more concerned with the business strategy, by focusing on change requests that are at the top of the task prioritization in the organization [52], enabling autonomy over authority when it comes to making decisions regarding changes [21]. The need of a change manager to manage a backlog of changes, which was also responsible for managing the priorities and together with the CAB, decide the priority on what would bring more business value, achieving organizational efficiency [53]. So, the CAB would act as “flight control” [1], i.e. coordinating different teams and tailoring their decisions to their business needs, and not being perceived as an entity to stop changes through bureaucratic processes.

Table 2. *DevOps and Change Management Studies in the Literature*

ID	Article	Concept						
		Automation	**CI / CD**	**Metrics and Monitoring**	**Cultural Shift**	**Communication**	**Task Prioritization**	**Strategic CAB**
1	[19]	●	●		●			●
2	[43]	●						●
3	[53]						●	●
4	[52]						●	
5	[44]	●						
6	[42]	●	●					
7	[45]	●	●	●	●			
8	[54]	●	●	●				●
9	[55]	●	●		●		●	
10	[56]	●					●	
11	[57]			●	●	●		
12	[49]	●	●		●			
13	[48]	●			●			
14	[1]	●	●					●
15	[40]	●	●		●			
16	[58]			●		●	●	
17	[59]				●			
18	[60]	●						
19	[21]	●		●	●			●
20	[50]		●	●				
21	[41]	●		●	●	●		
22	[2]		●					
23	[61]	●						
24	[46]	●		●				
25	[51]	●				●		●
26	[62]					●		
27	[47]	●		●				

3.5. Summary

Overall the studies demonstrate that despite the challenges associated to the adoption of DevOps and the integration of CM, they can coexist. It is important to note that both are complimentary and share the ultimate goal of produce business value. Whereas CM aims to control and mitigate risk, DevOps aims for speed up development backed by automation fostered by a collaboration culture. The four key findings found in the literature that sum up the integration of CM and DevOps are:

Automation is the key and also serves the purpose of mitigating the bureaucracy in CM, besides the already know advantage of speeding up development.

Communication is essential across teams regardless the approach, however, by having a unified channel where information can be exchanged and accessed is efficient and fosters collaboration.

Having incremental changes it is easier to trace eventual issues, which enables the CM process to be streamlined; furthermore, staged deployments on canary environments guarantee that if an issue arises, the entire deployment is not compromised and a fix can be done immediately.

The CAB can delegate some decisions (depending on their priority) to another team or team member, enabling autonomy; acts as a "flight controller" and is deeply business driven when a decision regarding a change have to be done.

Despite being a fruitful discovery, none of the presented studies explored such relationships in depth backed by a case study but presented a set of guidelines in their studies. Hence, as stated in motivation in the Introduction chapter, this research aims to explore the relationship between DevOps and CM. Since there is no research about DevOps applied in the CM process, this leads us to conclude the exploratory aspect of this study.

CHAPTER 4

Research Methodology

In this chapter, the author introduces how the research will be conducted in order to understand the connection between DevOps and CM. To accomplish this goal, the author planned a case study to unveil the answers stated by the research question.

4.1. Designing the Case Study

Figure 2 illustrates the CS design, and in the course of this section the reason for this design will be discussed.

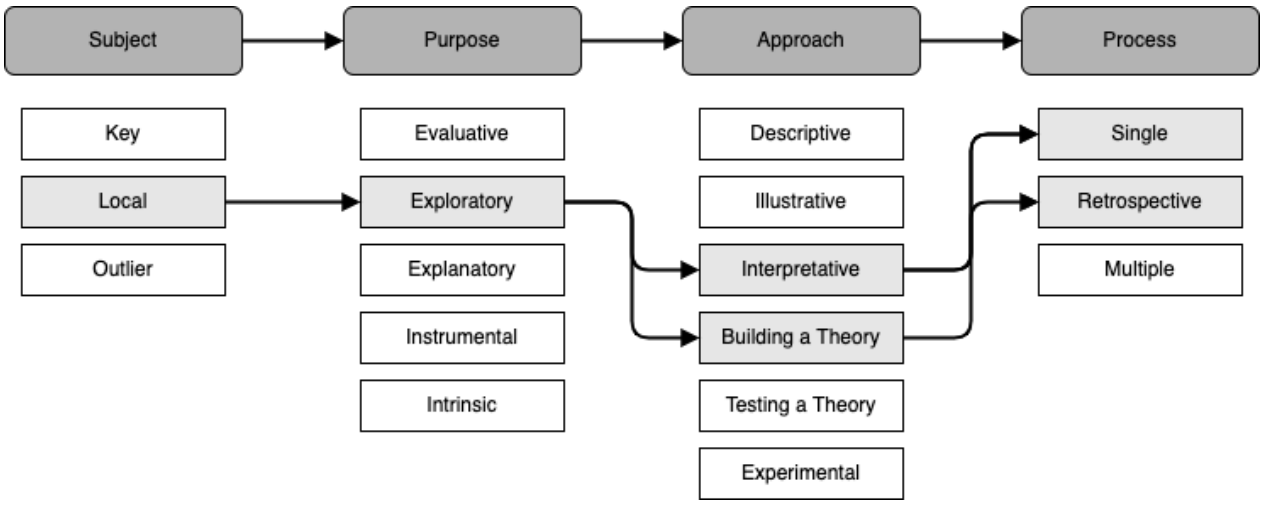

Figure 2. *Methodology approach by Thomas, adapted from [63]*

First the *Subject*, i.e. the focus for this study, which is critical as it explains the research methods [63]. As the author works on a team that partially applies the topics of this research, this CS can be classified as a local knowledge case.

A CS revolves around a *Purpose* [63], which in this study is represented by the main question "*What is the relationship between DevOps and Change Management?*". As a result, this question

develops into four RQs to be examined in this study. Table 3 details the set of RQs for this investigation. As there is a "what" question, it leads to inquiry further reinforcing the exploratory nature of this study [64]. Given the interest in understanding this phenomenon, it leads to the conclusion of a exploratory nature of this study.

Table 3. *Research Questions*

Research Question ID	**Description**
RQ1	Which DevOps practices can be used in the Change Management Process?
RQ2	What are the benefits of using DevOps practices?
RQ3	What are the challenges of using DevOps practices?
RQ4	How DevOps can help when there is a change?

In regard to the *Approach*, the CS will focus on two targets:

- Building a Theory — as observed in the literature review, there was no evidence of the relationship between DevOps the CM process.
- Interpretative — this is intended to broaden our understanding about DevOps and the CM process, and how they can be related.

This research has no initial theory to test, nonetheless grants us the opportunity to build a new theory for further studies based on the ideas that may emerge as a result of its conclusions. Furthermore, it intends to deepen the understanding of how changes can be managed with DevOps and highlight which practices can be related to CM.

Regarding the *Process*, i.e. how the CS will be conducted [63]: a single team will be the subject of the investigation, therefore a single unit of analysis [64].

Finally, time is an relevant factor to consider [63], and for this study the contemplated time-frames are:

- Retrospective — where the data collection relates to a previously occurring phenomenon;
- Snapshot — when the time occurs in a given time-frame such as a month, week, day or another period of time.

This study focuses on the potential relationship between DevOps and the CM process based on the (past) experience of a single team, which corroborates its retrospective quality. The major data collection is predominately document analysis and interviews of practitioners who experienced the phenomenon.

In a nutshell, the presented CS will be a single, retrospective approach to answer our RQs. For the purpose of supporting the interview process, a questionnaire was developed to assist the author during the CS. For reference, the selected DevOps practices for the questionnaire were extracted from [65] work, and the CM from the ITIL reference book [37].

4.2. Conducting the Research

This research was conducted over the course of 2 months (August and September 2022) at a multinational *fintech* company, with one team and a sample of 10 interviewees, for a qualitative approach.

The members of the team in this study are of diverse linguistic backgrounds, therefore each interview was conducted in both English and Portuguese, languages native to the interviewer and most interviewees. The questionnaire has a fixed number of questions with the following structure: (i) professional experience and role; (ii) DevOps experience; and (iii) DevOps and Change Management.

To ensure that the recorded interviews were understood, each one was reviewed, and the relevant topics were transcribed for written support using the same terminology observed in the recordings, with the authorization of all participants. The conclusions were translated by the author into English, as it reflects the language of this study.

Finally, the team structure is summarized in Table 4, detailing each Interviewee's role and professional experience.

Table 4. *Interviewees Details*

Interviewee	**Position**	**Experience (Years)**	**Experience in DevOps (Years)**
I_1	Quality and Assurance (QA) Engineer	15	0
I_2	Quality and Assurance (QA) Engineer	12	2
I_3	Software Engineer	1	1
I_4	Software Engineer	10	2
I_5	Software Engineer	3	1
I_6	Software Engineer	7	1
I_7	Software Engineer	2	1
I_8	Technical Writer	7	0
I_9	User Experience (UX) Designer	6	0
I_{10}	Product Manager	20	0
Average	—	8.3	0.8

The team's average professional experience is 8.3 years, and experience with DevOps is about 0.8 years. It is important to note that for the CM experience, most participants were neither fully aware of the concept nor had knowledge about the process as a whole. So, to address this knowledge gap, during the interviewing process, the interviewer explained the main concepts behind DevOps in addition to CM and its phases. This way, such initial engagement would help the participants to complement their answers in the questionnaire. Despite their limited experience with both experiences, their perspective as practitioners in the field are valuable for this CS.

CHAPTER 5

Case Study Analysis and Reporting

The first section of the questionnaire consists of a few fundamental questions about DevOps. This section has three main goals: (i) understand how our interviewees perceived it conceptually; (ii) inquire how one can benefit from this culture; and finally, (iii) what are the challenges one encounters when adopting its respective practices. The second point serves to answer our RQ2, whose detailed explanation is unveiled in Section 5.2. Moreover, to enhance the findings in RQ2, the scope of the inquiry was extended by asking our interviewees which practices they had the opportunity to work with, and provide their rationale regarding the benefits. Regarding the challenges (Section 5.3) addressed by RQ3, based on each practice it was surveyed how hard would be to implement them alongside the reasons behind it.

The second section aims to disclose, based on our interviews, which DevOps practices are related to CM, extending the scope to each type of change: *standard*, *normal* and *emergency*. Our interviewees were also asked their opinion on how a change can benefit from DevOps. The answers to RQ1 and RQ4 are explained in Section 5.4 and 5.5, respectively.

The findings for this research and answers to the RQs stated in the previous chapter will be discussed in the following sections.

5.1. What is DevOps?

After gathering the professional experience of each participant, the questionnaire began with the following questions: "*How familiar are you with DevOps?*" and followed with "*How would you describe DevOps in your own words?*". It was intended to understand the interviewees' background regarding the core concepts behind this methodology and additionally, promote an initial engagement about one of the topics of this research.

The first question ranked the familiarity of each interviewee with their perceptions of DevOps on a scale from 1 to 5, meaning "Not Familiar", "Slightly Familiar", "Neutral", "Familiar" and

"Very Familiar", respectively. For this team, the average knowledge yielded a value of 3.5. To illustrate the distribution among all participants, Figure 4 depicts the team knowledge distribution.

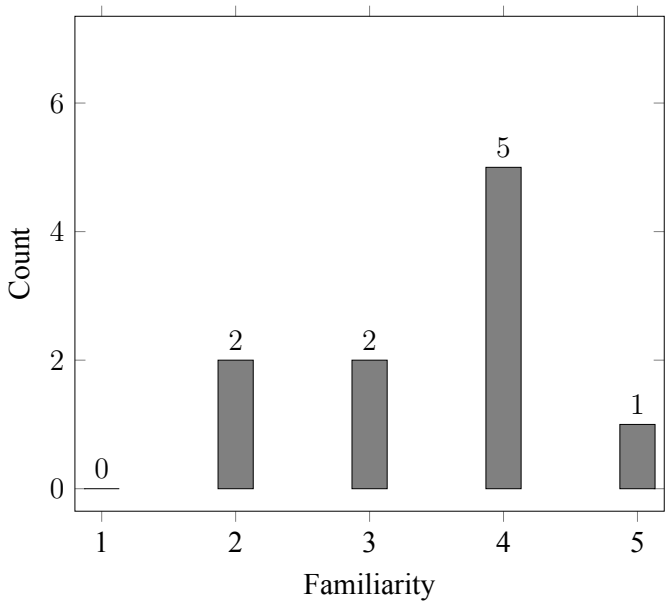

Figure 3. Interviewees' DevOps Familiarity Distribution

In the next question, the contributors were asked to describe DevOps by using their own words. After analyzing each team member's answers, which can be found in Appendix A, it is possible to interpret it as: "*an agile and process-driven agile methodology that aims to speed up SDLC and deliver business value. It is enabled by automation, which ensures faster delivery to its end users. Also, it furthermore reinforces collaboration between different team members during software development.*

Overall, the team is considerably aware of this concept, more specifically on how it enhances the entire SDLC and the developer experience, implying that it can be used to join the many team members (software development and operations). This finding suggests a validation of what can be found in the literature, which considers this collaborative culture as a part of its fundamental principles [65]. However, this seems contradictory considering their experience presented in Table 4. This can be explained by most participants having worked with more practices on average during their professional tenure. This is supported by Table 5 below, which summarizes the results from the question "*Which DevOps practices below have you ever worked with?*", with a scale from 1 to 2, meaning "Never Worked" and "Already Worked", respectively.

Table 5. DevOps Practices Interviewees Worked With

Practices	Rate	
	1 - Never Worked	2 - Already Worked
Continuous Planning	2	8
Feedback Loops Between Dev and Ops	3	7
Continuous Monitoring	4	6
Measure Performance Metrics (CI, Test and Ops)	4	6
Automated Feedback for Performance Models and Performance Predictions	9	1
Application Monitoring	5	5
Automated Dashboards	6	4
Continuous Integration	2	8
Prototyping Application	5	5
Continuous Deployment	4	6
Automated Deployment	4	6
Continuous Delivery	3	7
Continuous Testing	4	6
Automated Testing	2	8
Process Standardization	5	5
Infrastructure as Code (IaC)	9	1
Stakeholder Participation	2	8
Average	6.3	9.7

5.2. DevOps Benefits

In order to grasp an overview of the advantages of DevOps, itself an explanation of RQ2, the questionnaire included the open-ended question *"Can you describe its potential benefits?"*. Based on Appendix B, where the answers are fully reported, it is possible to highlight two beneficial outcomes of DevOps: (i) a better product, and (ii) faster software delivery. In the first case, the enhancement

of communication between the team and other stakeholders contributes to the success of the project. All team members (developers, product managers, and end users) are involved in the development process. The second outcome can be explained from a team-based development perspective: it is achieved by the team adopting a methodology in which the developer experience as a whole is improved and insured by a DevOps process employing automation tools.

To delve further into this RQ, the author aims to examine the benefits by practice. In so doing, the introductory question *"Which DevOps practices below do you know?"*, whose answers are translated into Table 6 where each known practice per interviewee is summarized.

Table 6. *DevOps Practices Known by Interviewees*

	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}	**Total**
Continuous Planning		●	●		●	●	●	●	●	●	8
Feedback Loops Between DevOps	●	●	●		●	●	●	●			7
Continuous Monitoring	●		●	●	●	●		●		●	7
Measure Performance Metrics (CI, Test and Ops)	●			●	●	●	●			●	6
Automated Feedback for Performance Models and Performance Predictions					●						1
Application Monitoring	●		●	●	●	●		●		●	7
Automated Dashboards		●		●	●	●				●	5
Continuous Integration	●	●	●	●	●	●	●	●		●	9
Prototyping Application			●		●	●	●			●	5
Continuous Deployment	●	●	●	●	●	●	●	●		●	9
Automated Deployment	●	●	●	●	●	●	●	●		●	9
Continuous Delivery	●	●	●	●		●	●		●	●	8
Continuous Testing	●	●		●	●	●	●		●	●	8
Automated Testing	●	●	●	●	●	●	●	●		●	9
Process Standardization	●					●		●	●	●	5
Infrastructure as Code (IaC)						●					1
Stakeholder Participation	●	●	●		●	●	●	●	●	●	9
Average	65%	53%	65%	59%	82%	94%	59%	59%	29%	82%	6.6

Table 6 demonstrates that the team is familiar with almost half of the practices, knowing on average 6.6. The median is 7, complementing the average. The least known practices are *Automated Feedback for Performance Models* and *Infrastructure as Code (IaC)*.

After analyzing each answer from "*What are the benefits you consider relevant when using one of the practices mentioned in the previous question? Why?*", the author compiled the benefits in Table 7, based on the full answers available in Appendix C. Some practices were omitted due to a lack of data.

Table 7: *DevOps Benefits per Practice*

Practice	Conclusions
Continuous Planning	It is fundamental, or as an interviewee described "*it is the foundation of everything*". Employing good planning contributes to the product vision, i.e. where the team is heading in terms of what needs to be delivered. The quality is an outcome of how the requirements were aligned (expectation management) and incorporated in the process. Also, it empowers the team to react to changes and/or new requirements that either come from the stakeholders or during the development cycle, creating this effective communication-oriented practice.
Feedback Loops Between Dev and Ops	It contributes to continuous improvement among team members. It serves to assess "*what went wrong?*" and "*what went right?*". It is also a means to facilitate communication of the lessons learned for each iteration.
Continuous Monitoring	It helps to understand the system performance in production and to provide information so as to be able to react sooner rather than later. This way, the system achieves stability and the team can profit from the information to make an action plan whenever an incident arises.
Measure Performance Metrics (CI, Test and Ops)	It identifies the current application state in production and verifies performance bottlenecks. It complements other metrics, such as key performance indicators (KPIs), operational and delivery flow, and is quite useful to react to changes when they arise. This way, just like Continuous Monitoring, it is advantageous to be reactive to such metrics.

Application Monitoring	Just like in Continuous Monitoring, it bears valuable information to the team in such a way that it can be proactive, instead of reactive, when handling incidents. This way, the team can work on a strategy to tackle an issue before it affects the end users.
Automated Dashboards	It summarizes the information graphically instead of in a tabular manner, and checks the system performance. This way, by adopting such monitoring tools, the team can react swiftly based on graphical information and to combine it with other key performance indicators (KPIs).
Continuous Integration	It draws on two beneficial aspects: product delivery and developer experience. The first enables fast, reliable and predictable software that is deliverable in the development process. The second suggests improving the focus on development tasks, making the concerns about software integration agnostic to the developers.
Prototyping Application	It introduces the product vision and strategy to the team and stakeholders earlier, which helps to evaluate the concepts of the new end product. Further contributing to product improvement, it may impact product planning based on findings from the prototype as potential issues are unveiled at earlier stages in product development .
Continuous Deployment	Given its continuous nature, it provides fast deployment to the end users as soon as they are meet the quality requirements.
Automated Deployment	As it is a process ensured by automation, it promotes predictable and deterministic deployments. Manual interventions may be error-prone, and when adopting an automated approach, the risk of failure is reduced.

Continuous Delivery	It reduces time and effort when delivering software, whether improvements/new features, or fixing bugs.
Continuous Testing	Considering that any new change will be tested, it is a way to minimize the impact on existing production implementation by changing/adding new code. In effect, it functions as a quality gate that ensures the system is consistent in its functioning. This is backed by tests, which provide a certain confidence level based on the robustness and certainty given by this quality gate.
Automated Testing	It serves as quality guarantee on the deliverable, as for each new introduced change, given the tests' deterministic nature, increasing the odds of shipping non breaking features, which creates a robust and safe deliverable.
Process Standardization	It facilitates the process for different team members, ensuring deterministic outcomes, because the process will be followed thoroughly and, as a result, it reduces cognitive load as it is something well-known among team members.
Stakeholder Participation	By keeping different stakeholders "*in the loop*", they are able to contribute to the vision of what is expected and foment knowledge sharing between different parties. Additionally, it supports the decisions made during planning. Therefore, it enables a collaborative and communicative environment between everyone in the team, including product and engineering stakeholders.

Based on the conclusions listed in Table 7, it is possible to attest that some practices are synergetic. For instance, Continuous Planning benefits positively when using metrics provided by monitoring driven practices, such as Continuous Monitoring and Stakeholder Participation. Furthermore, during the interview process, the author observed that the majority of participants grouped some practices into a single concept, and identified the same reasons behind the benefits of the given practice. This suggests that to secure the most benefits from DevOps practices, one should try to combine at least a set of practices, rather than choose them individually. Therefore, by combining a set of them, one can get the most out of it, which is corroborated by the participants' perspectives.

Quality assurance is another important aspect we can extract from some of the practices, such as Continuous Testing and Automated Testing. This is achieved given its deterministic nature, meaning it provides a confidence level to what is being deployed will reduce possibility of errors in production.

Faster delivery is also implied by Continuous Deployment and Continuous Delivery, given that they contribute to faster deployments, due to the automation they are built on, so they are available as soon as possible to their end users. Moreover, it contributes to the quality requirements being met, as manual intervention is reduced. This indicates another way of guaranteeing quality assurance and an example of the synergy between practices.

These last two mentioned benefits extracted from from Table 7 relate to an in-depth study [66] about DevOps benefits. For instance, the quality assurance refers to "Increase of code quality" and the faster delivery to "Faster time to market", meaning the conclusions are aligned with other researchers.

5.3. DevOps Challenges

To determine the DevOps challenges, the author began inquiring "*How hard do you think it would be to implement these practices?*" with a scale from 1 to 5, meaning "Easy", "Very Easy", "Neutral", "Hard" and "Very Hard", respectively. The main goal was to classify the difficulty of each practice's implementation based on the participants' perspectives. For the practices which had no knowledge about them, the selected answer was "Neutral". This is important to note because the sums and averages from practice to practice may vary. The compilations for each practice are summarized in Table 8.

Table 8: DevOps Difficulty per Practice

Practices	Rates					
	1 - Very Easy	2 - Easy	3 - Neutral	4 - Hard	5 - Very Hard	Average
Continuous Planning	2	4	1	3	-	2.5
Feedback Loops Between Dev and Ops	2	5	3	-	-	2.1
Continuous Monitoring	1	3	4	2	-	2.7
Measure Performance Metrics (CI, Test and Ops)	-	2	3	4	1	3.3
Automated Feedback for Performance Models and Performance Predictions	-	-	8	1	1	3.3
Application Monitoring	2	2	4	2	-	2.6
Automated Dashboards	2	2	5	1	-	2.5
Continuous Integration	-	3	3	3	1	3.2
Prototyping Application	1	3	5	1	-	2.6
Continuous Deployment	-	2	1	5	2	3.7
Automated Deployment	-	-	1	7	2	4.1
Continuous Delivery	-	2	2	6	-	3.4
Continuous Testing	-	5	2	2	1	2.9
Automated Testing	-	4	5	1	-	2.7
Process Standardization	-	3	5	2	-	2.9

Infrastructure as a Code (IaC)	-	1	6	2	1	3.3
Stakeholder Participation	3	3	1	2	1	2.5

To complement the reasons behind each participant's answer, it was intended to justify their assessment and gather their opinion on each specific practice. The comments for each practice are available in Table 9 and the detailed table can be seen in Appendix C. The categorization of each practice is set to either *easy* or *hard*, by grouping the lower and upper limits into one. The outcome was based on the average value per practice. In case of a tie, the count per practice by limit (easy or hard) was applied to make a decision.

Table 9: *DevOps Practices Comments on Adoption Difficulty*

Practice	Comments
Continuous Planning	It tends to be **easy**, due to its human component, and can be incorporated in the daily team activities, but it is relevant to note that it depends on the team maturity. Nevertheless, it depends on the business context, i.e. continuous planning can be challenging to achieve if the company's vision changes rapidly.
Feedback Loops Between Dev and Ops	It tends to be **easy**, due to its human component and it is a matter of organization among the team members.
Continuous Monitoring	It tends to be **easy**, because there are tools that are ready-to-use and the challenge lies in knowing what is relevant to be monitored. On the other hand, some participants believe it requires human and technical effort to perform such a complex task.
Measure Performance Metrics (CI, Test and Ops)	It is **hard**, given the complexity of its implementation and in-depth systems knowledge. Furthermore, it is not able to confer results promptly, as the time needed to collect relevant data and it be useful can be long.
Application Monitoring	Similarly to Continuous Monitoring, it is **easy**, due to the available tools, despite sharing the same challenges stated to its monitoring practice counterpart.
Automated Dashboards	Similarly to Continuous and Application Monitoring, it is **easy**, due to the available tools.
Continuous Integration	It tends to be **hard**, as it is dependent on business context and requires technical effort for its development. Some participants pointed out that some tools already have this, reducing the technical effort, making this practice relatively easy to implement.

Prototyping Application	It is **easy**, as it is only a prototype and this draft does not need to be a complete feature.
Continuous Deployment	It is **hard**, as it is complex to attain and inspect, e.g. a "*black box*", and during the deployment phase there are nuances that make it hard to implement.
Automated Deployment	It is **hard**, because it relies on nuances in the process, the technical effort and its potential to introduce new changes to the code base that might not be ready.
Continuous Delivery	It is **hard** similarly to Continuous and Automated Deployment, because the team must ensure the system will work when new changes are delivered, for instance backwards compatibility.
Continuous Testing	It tends to be **easy** for most participants given the existence of tools and being something that can be performed in the early stages of the SDLC. But as some pointed out, it can be hard depending on the size of the scope to be retested. Running all tests is not feasible with the difficulties inherent in maintaining such continuity.
Automated Testing	It is **easy**, due to be something that can be easily incorporated in the SDLC.
Process Standardization	It is **easy**, due to its human component and it is a matter of organization among the team members.
Stakeholder Participation	It tends to be **easy**, as it requires communication and dedication from everyone to include all the stakeholders in the process. However, the dedication itself can be a downside to this practice, as the stakeholders must be integrated in the process and it must be of relevance. Additionally, effective communication among different teams/stakeholders may not be straightforward.

At a first glance, this approach seems to answer the proposed RQ regarding the DevOps challenges. Nevertheless, it only presents the difficulties (and the facilities) to implement the listed practices. By exploring the comments on Table 9 and the answers in Appendix C, it is possible to unveil latent challenges that some practices pose.

An interesting aspect one can extract from the comments is how communication plays a substantial role in the practices. For some participants, Continuous Planning can be easier because it relies on communication (I_2 and I_7) and can be incorporated in the team's daily work (I_2 and I_6). Nevertheless, this research demonstrates that it can actually be difficult to achieve. For instance, how effectively is the message disseminated amongst all team members (I_5 and I_{10}). Another relevant aspect that plays an influential role is the business environment and its maturity (I_2 and I_8). Still, within the scope of communication as well as getting the message across, and regarding Stakeholder Participation, it calls for alignment moreso when it needs customer involvement, besides the effective communication (I_2, I_5 and I_9).

Regarding the practices that involve technical expertise, one can generally observe that monitoring practices (Continuous Monitoring, Application Monitoring and Automated Dashboards) are more easily adoptable, given the tools to supports its adoption (I_1, I_4 and I_9). On the other hand, some (I_1 and I_6) challenge such effortless implementation by pointing out the technical challenges it entails, as well as the know-how these practices require to provide relevant information. This indicates that despite having ready-to-use tools, how to implement them is an underlying challenge that cannot be neglected.

In relation to the technical effort, Continuous Delivery, Automated Deployment and Continuous Deployment share the same concern (I_1, I_3, I_4, I_5), but because they are part of the software development process, they introduce complexity in the deployment stage (I_6 and I_{10}). The difficulty regarding these practices revolve around their implementation (I_1) and more specifically, the pipelines to ensure its functioning (I_7). Furthermore, having many steps, it requires quality mechanisms to avoid introducing malformed deployments and ensure that all edge cases are covered within these steps (I_6). Something to note is the business and its setup (I_7 and I_9). This implies that the technical challenges for these practices present another layer to the implementation itself, but also the organizational context and the nuances it encompasses.

5.4. Change Management and DevOps

For this RQ, the questionnaire started with the same strategy to rank interviewees' familiarity regarding CM. For this team, the average knowledge yielded a value of 1.9. To illustrate the distribution among all participants, Figure 4 depicts the team knowledge distribution.

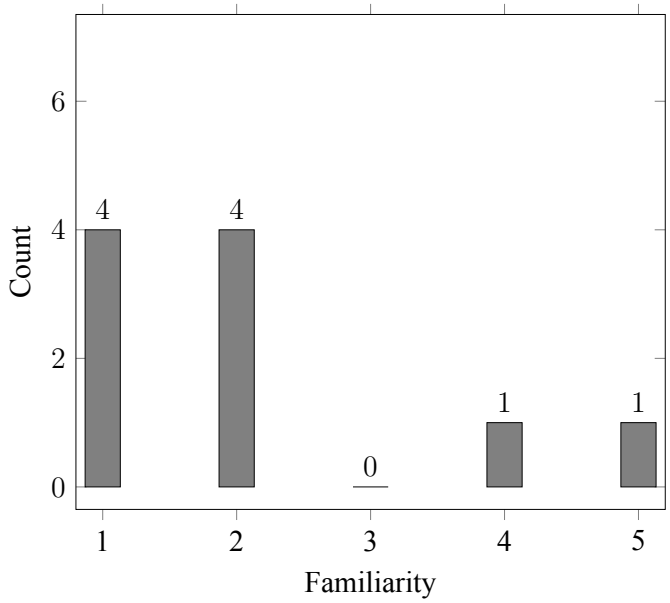

Figure 4. Interviewees' Change Management Familiarity Distribution

Figure 4 reveals that most of the team are not familiar with the CM process, meaning that this RQ will be challenging. The following subsections will examine the potential relationship between DevOps practices and the CM process for each type of change (standard, emergency and normal). For each type, the question "*Which one of the DevOps practices below applies to the Change Management Process?*", followed by "*Could you provide why and how these practices can be related?*" will be explored.

5.4.1. Standard Changes

The focus will be on standard changes and the DevOps practices it can be related to. Table 10 summarizes the information collected from the interviewees and serves as a foundation to discuss the practices. The table lists only the practices that met the criteria: select only practices that occur at least 5 times (half of the team) in a CM phase and are supported by the participants' answers. The tables with the summary of the Appendix D contains the tables for each change classification.

Table 10. Standard Change and DevOps Practices Distribution

Practices	Change Management Stage								
	Create RFC	**Record RFC**	**Review RFC**	**Access and Evaluate Change**	**Authorize Change and Build Test**	**Coordinate Change and Build Test**	**Authorize Change Deployment**	**Coordinate Change Deployment**	**Review and Close Record**
Continuous Planning	3	1	3	8	1	2	-	3	1
Continuous Integration	5	4	3	3	5	5	4	4	1
Continuous Deployment	4	3	3	3	3	4	5	7	3
Automated Deployment	4	3	3	3	3	4	5	8	3
Continuous Delivery	4	3	5	4	4	5	4	6	5
Continuous Testing	3	2	6	3	3	5	1	2	2
Automated Testing	2	2	5	2	4	5	1	2	1
Process Standardization	4	3	5	3	2	4	2	3	4
Stakeholder Participation	5	2	3	2	-	1	1	2	4

Based on Table 10 and the interviewees' answers (Appendix E), the author proposes a table where there is a mapping between the practice and the CM phase. The discussion of the findings is given in Table 11 below.

Table 11: DevOps Practices and Standard Changes Relationship

Practices	**Conclusions**
Continuous Planning	It is in "*Access and Evaluate Change*" that this practice is suggested to be more relevant, because it is an opportunity the team has to assess the value, impact and further evolution of this change (I_2, I_7 and I_{10}).
Continuous Integration	Overall, except for the "*Review and Close Record*", it highlights the authorization and coordination steps, since validation and authorizations are needed whenever there is change (I_1, I_2, I_5).
Continuous Deployment	The emphasis was mostly on "*Authorize Change Deployment*" and "*Coordinate Change Deployment*", because changes need to be validated and authorized to be deployed into the system (I_1, I_2, I_3, I_4 and I_7).
Automated Deployment	Similar to Continuous Deployment, the changes need to be validated and authorized to be deployed into the system. (I_1, I_3, I_7).
Continuous Delivery	It relates the "*Review RFC*" and "*Review and Close RFC*". Furthermore, coordination steps, namely "*Coordinate Change and Build Test*" and "*Coordinate Change Deployment*" because software is being delivered and coordination must be ensured, so that it does not cause any issues to the system (I_1, I_9 and I_{10}).
Continuous Testing	It needs to be part of the requirements, e.g. during the "*Review RFC*" and also the coordination and building of the test (I_2 and I_{10}).

Automated Testing	Similar to "*Continuous Testing*", it can be related to "*Review RFC*" as being part of the early requirements and the need to the coordination of this change deployment (I_2 and I_{10}).
Process Standardization	This practice is evenly distributed among all practices as "*everything in a process must be taken into consideration*", meaning the majority of phases can be related to this practice (I_9 and I_{10}).
Stakeholder Participation	The stakeholders are mostly involved in the creation "*Create RFC*" and the closing of the change request "*Review and Close Record*", because they can be part of this change, making sure they aware of its introduction, and optimizing communication. (I_8, I_9, I_{10}).

Based on the Table 11 above, it is possible to observe a pattern among "Continuous Integration", "Continuous Deployment", "Automated Deployment" and "Continuous Delivery", where coordination and authorization play an important role. The main reason is related to the need of coordination and authorization of the change. This can be verified in Appendix E, where some participants' answers grouped practices into one, and gave the same reason why they are related to the CM steps. A similar phenomenon happened with "Continuous Testing" and "Automated Testing" where the same association was made.

5.4.2. Emergency Changes

For emergency changes, the same approach was undertaken in subsection 5.4.1. The summarized information for this category is represented by Table 12 below. The practices selection utilized the same criteria defined in Section 5.4.1.

Table 12. Emergency Change and DevOps Practices Distribution

Practices	**Change Management Stage**						
	Review RFC	**Access and Evaluate Change**	**Authorize Change and Build Test**	**Coordinate Change and Build Test**	**Authorize Change Deployment**	**Coordinate Change Deployment**	**Review and Close Record**
Continuous Planning	4	6	3	4	2	3	3
Continuous Integration	3	4	5	5	3	4	2
Continuous Deployment	3	4	3	4	5	6	3
Automated Deployment	3	4	3	4	4	7	3
Continuous Delivery	4	5	4	4	3	5	3
Continuous Testing	1	3	4	5	-	2	-
Automated Testing	1	4	6	6	1	3	1
Infrastructure as a Code (IaC)	2	1	2	2	1	2	1
Stakeholder Participation	5	3	3	4	2	3	5

Table 12 yielded the same practices as introduced in the previous section. In terms of CM, most interviewees chose the same mapping they did in the previous section. To support these claims

(detailed in Appendix E) some commented that this would be "the same" process. Still, following this lead, they highlighted the importance of priority for this type of change. This indicates that for emergency changes, on top of the conclusions gathered previously in Subsection 5.4.1, the reason behind the selected practices related to the CM process would be the same. In addition, ensuring that the authorization steps would be followed was also important. Therefore, this suggests that the process the change should follow must be ensured regardless of its related practices. At the same time, priority should be taken into consideration, as it is influential in this scenario (I_2, I_3, I_6). Furthermore, there should be an emphasis on authorization for these changes in such critical scenarios, because given an emergency the system should not have been validated thoroughly and be quality-assured (I_4, I_5 and I_7).

5.4.3. Normal Changes

Normal changes had the same practices as the two other categories related to CM process, and are detailed in Table 13. The practices selection utilized the same criteria defined in Section 5.4.1.

Table 13. Normal Change and DevOps Practices Distribution

Practices	**Change Management Stage**						
	Review RFC	**Access and Evaluate Change**	**Authorize Change and Build Test**	**Coordinate Change and Build Test**	**Authorize Change Deployment**	**Coordinate Change Deployment**	**Review and Close Record**
Continuous Planning	5	5	4	3	1	5	3
Continuous Integration	2	4	6	5	3	4	2
Continuous Deployment	2	4	3	4	4	6	2
Automated Deployment	1	3	2	3	3	6	1
Continuous Delivery	5	6	5	5	4	6	4
Continuous Testing	2	5	5	5	1	3	3
Automated Testing	3	5	5	5	1	3	3
Process Standardization	5	4	2	4	1	2	4
Stakeholder Participation	5	5	2	5	-	2	6

By inspecting Appendix E and the Table 13 above, in addition to the remarks identified in Table 11, the author identified a subtle difference for this type of change. Contrary to what was observed in Subsection 5.4.2, authorization for the deployment of the change did not seem to be as significant as its emergency counterpart. As a contributor pointed out, "*it is not urgent and does not need to be escalated*" (I_2), implying it could bypass the authorization phase, as the main goal is to ensure quality when delivering this change (I_4 and I_5).

5.5. DevOps and Change

In order to unveil the last RQ, the questionnaire concluded with the open-ended question *How do you think DevOps is helpful when there is a change?*, aimed to grasp a common perception between DevOps and Change. By interpreting the answers in Appendix F, it is possible to verify across each contribution just how substantial DevOps is when a team faces a change to be addressed. This is supported by how automated processes promote faster deliveries (I_2, I_5 and I_6), meaning rapid delivery to the end users as a result of this *continuous* approach (I_8). Automation ensures both quality gate (I_3 and I_3) and traceability (I_7). Finally, on the team level, it promotes autonomy and readiness to implement those changes (I_1), as well as collaboration among all stakeholders and the team itself (I_8, I_9 and I_{10}).

CHAPTER 6

Conclusions

This research aimed to explore an area that has not yet been explored before, and contribute to the academic and scientific community. For this study, a set of data was collected from interviews conducted with IT professionals who apply DevOps practices, and deal with changes in their daily work.

Based on each participants' experiences, it is possible to conclude that there is a relationship between some of the DevOps practices and the CM process. This includes the software delivery process, from development (and testing), integration, to deployment. Also, Continuous Planning and Stakeholder Participation were significant, especially in the review and closing stages, due to their involvement in the beginning and end of the process. Regarding change delivery, this research shows that it must performed thoroughly and ensure quality when deploying, bearing in mind how critical it is that its proper prioritization is defined. In regards to the concept of change and how DevOps can be useful, the team pointed out how it provides a way to rapidly react to change. Furthermore, when the change is addressed, the existing automation prevents the introduction of errors in the production environment.

It was possible to witness the practitioners' perspectives on how DevOps is intrinsically beneficial when facing a change. Overall, the team highlights the faster delivery as a substantial benefit, alongside the alignment it brings due to an inherently collaborative environment. Also, the practices themselves complement each other, i.e. adopting just one does not unleash all the potential benefits DevOps can bring, indicating the close relationship that they have.

This research features the associated challenges when adopting some of its practices. Despite communication being part of the DevOps culture, this study demonstrates that communication poses a challenge between different team members and even as high as the company level. Another challenge noted is the technical knowledge required to implement infrastructure such as pipelines, which guarantees the automated processes will deliver software to the end users.

Still, this study presented many limitations that must be discussed. The number of participants and interviews is small, and the professional experience of the participants might have played a role in the outcomes. Given all participants' work in remote-friendly positions, it was not possible to have personal engagement with each one of them. Therefore, direct observation of the team working together was not possible.

The team is well-versed in DevOps practices and knows the importance of their application, but in the CM counterpart, most of them were not familiar with the process; only a few of them had experience with it. This clearly has an impact in the outcome of the research question. For RQ1, most participants found difficulties in trying to map the practices to the CM process. This means that most of the answers lacked granularity in their rationale, being relatively superficial why those practices would be related to the DevOps process. Another limitation is related to the exploration the author proposed on how such practices would be implemented. Most of these constraints seem to be rooted in their lack of deep experience in CM, with which most of them had merely a few years of experience. Some of the interviewees made it clear that they did not have broad knowledge of CM, despite having work experience therein.

Finally, the author believes that further research has the potential to add depth to the findings. Further work could investigate how DevOps practices can be applied to other processes of the ITIL framework.

Bibliography

[1] S. D. Mack, *Adaptive change management: A devops approach to change management*, Feb. 2020. [Online]. Available: https://dzone.com/articles/adaptive-change-management.

[2] S. RADHAKRISHNAN, *Integrating itil change management and devops*, Mar. 2018. [Online]. Available: https://devops.com/integrating-itil-change-management-and-devops/.

[3] H. Holmström, B. Fitzgerald, P. Ågerfalk, and E. Conchúir, "Agile practices reduce distance in gloral software development," *Information Systems Management*, vol. 23, no. 3, pp. 7–18, 2006, cited By 169. doi: 10.1201/1078.10580530/46108.23.3.20060601/93703.2.

[4] K. Beck, C. Andres, and E. Gamma, *Extreme Programming Explained: Embrace Change*, ser. XP series. Addison-Wesley, 2004, isbn: 9780321278654. [Online]. Available: https://books.google.pt/books?id=32RGBAAAQBAJ.

[5] K. Schwaber and M. Beedle, *Agile Software Development with Scrum*, 1st. USA: Prentice Hall PTR, 2001, isbn: 0130676349.

[6] D. Anderson, *Agile Management for Software Engineering*, 2004, cited By 117.

[7] D. Karlström and P. Runeson, "Integrating agile software development into stage-gate managed product development," *Empirical Software Engineering*, vol. 11, no. 2, pp. 203–225, 2006, cited By 93. doi: 10.1007/s10664-006-6402-8.

[8] B. Ramesh, L. Cao, and R. Baskerville, "Agile requirements engineering practices and challenges: An empirical study," *Inf. Syst. J.*, vol. 20, pp. 449–480, Sep. 2010. doi: 10.1111/j.1365-2575.2007.00259.x.

[9] M. A. G. Darrin and W. S. Devereux, "The agile manifesto, design thinking and systems engineering," in *2017 Annual IEEE International Systems Conference (SysCon)*, 2017, pp. 1–5. doi: 10.1109/SYSCON.2017.7934765.

[10] V. T. Heikkilä, D. Damian, C. Lassenius, and M. Paasivaara, "A mapping study on requirements engineering in agile software development," in *2015 41st Euromicro Conference on Software Engineering and Advanced Applications*, 2015, pp. 199–207. doi: 10.1109/SEAA.2015.70.

[11] S. Raza and U. Waheed, "Managing change in agile software development a comparative study," in *2018 IEEE 21st International Multi-Topic Conference (INMIC)*, 2018, pp. 1–8. doi: 10.1109/INMIC.2018.8595457.

[12] S. Sharma, D. Sarkar, and D. Gupta, "Agile processes and methodologies: A conceptual study," *International Journal on Computer Science and Engineering*, vol. 4, May 2012.

[13] G. Kim and I. R. Press, *Devops distilled, part 1: The three underlying principles*, 2013.

[14] M. Virmani, "Understanding devops amp; bridging the gap from continuous integration to continuous delivery," in *Fifth International Conference on the Innovative Computing Technology (INTECH 2015)*, 2015, pp. 78–82. doi: 10.1109/INTECH.2015.7173368.

[15] S. W. Hussaini, "Strengthening harmonization of development (dev) and operations (ops) silos in it environment through systems approach," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2014, pp. 178–183.

[16] J. Diaz, D. López-Fernández, J. Pérez-Martínez, and Á. González-Prieto, "Why are many business instilling a devops culture into their organization?," May 2020.

[17] J. Humble, J. Molesky, and B. O'Reilly, *Lean Enterprise: How High Performance Organizations Innovate at Scale*, ser. Lean series. O'Reilly Media, 2014, isbn: 9781491946558. [Online]. Available: https://books.google.pt/books?id=ivixBQAAQBAJ.

[18] R. M. D. Amaro, R. Pereira, and M. Mira da Silva, "Capabilities and practices in devops: A multivocal literature review," *IEEE Transactions on Software Engineering*, pp. 1–1, 2022. doi: 10.1109/TSE.2022.3166626.

[19] M. Ashley, *Change management in a fast paced devops world*, Dec. 2020. [Online]. Available: https://devops.com/change-management-in-a-fast-paced-devops-world/.

[20] M. Comuzzi and M. Parhizkar, "A methodology for enterprise systems post-implementation change management," *Industrial Management Data Systems*, vol. 117, pp. 00–00, Oct. 2017. doi: 10.1108/IMDS-11-2016-0506.

[21] I. Nyo, *Best practices for change management in the age of devops*, May 2020. [Online]. Available: https://www.atlassian.com/engineering/best-practices-for-change-management-in-the-age-of-devops.

[22] T. Dingsøyr, S. Nerur, V. Balijepally, and N. Moe, "A decade of agile methodologies: Towards explaining agile software development," *Journal of Systems and Software*, vol. 85, no. 6, pp. 1213–1221, 2012, cited By 546. doi: 10.1016/j.jss.2012.02.033.

[23] B. Henderson-Sellers and M. Serour, "Creating a dual-agility method: The value of method engineering," *Journal of Database Management*, vol. 16, no. 4, pp. 1–23, 2005, cited By 68. doi: 10.4018/jdm.2005100101.

[24] J. Highsmith and A. Cockburn, "Agile software development: The business of innovation," *Computer*, vol. 34, no. 9, pp. 120–127, 2001. doi: 10.1109/2.947100.

[25] T. Dybå and T. Dingsøyr, "What do we know about agile software development?" *Software, IEEE*, vol. 26, pp. 6–9, Nov. 2009. doi: 10.1109/MS.2009.145.

[26] S. Nerur and V. Balijepally, "Theoretical reflections on agile development methodologies," *Commun. ACM*, vol. 50, pp. 79–83, Mar. 2007. doi: 10.1145/1226736.1226739.

[27] T. Dingsøyr, T. Dybå, and N. Moe, "Agile software development: An introduction and overview," *Agile Software Development, by Dingsøyr, Torgeir; Dybå, Tore; Moe, Nils Brede, ISBN 978-3-642-12574-4. Springer-Verlag Berlin Heidelberg, 2010, p. 1*, vol. -1, p. 1, Apr. 2010. doi: 10.1007/978-3-642-12575-1_1.

[28] C.-Y. Hsieh and C.-T. Chen, "Patterns for continuous integration builds in cross-platform agile software development," *Journal of Information Science and Engineering*, vol. 31, no. 3, pp. 897–924, 2015, cited By 7.

[29] J. Kendall and K. KENDALL, "Agile methodologies and the lone systems analyst: When individual creativity and organizational goals collide in the global it environment," *Journal of Individual Employment Rights - J Indiv Employ Right*, vol. 11, pp. 333–347, Jan. 2004. doi: 10.2190/B6WV-TR2A-R42K-C87C.

[30] N. Kaleshovska, S. Josimovski, L. Pulevska-Ivanovska, K. Postolov, and Z. Janevski, "The contribution of scrum in managing successful software development projects," *Economic Development/Ekonomiski Razvoj*, vol. 17, no. 1-2, pp. 175–194, 2015, cited By 3.

[31] J. Humble and J. Molesky, "Why enterprises must adopt devops to enable continuous delivery," *Cutter IT Journal*, vol. 24, no. 8, pp. 6–12, 2011, cited By 101.

[32] C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano, "Devops," *IEEE Software*, vol. 33, no. 3, pp. 94–100, 2016, cited By 174. doi: 10.1109/MS.2016.68.

[33] G. B. O. of Government Commerce, *Planning to Implement Service Management*. The Stationery Office (TSO), 2002, isbn: 0113308779.

[34] W.-G. Tan, A. Cater-Steel, and M. Toleman, "Implementing it service management: A case study focussing on critical success factors," *Journal of Computer Information Systems*, vol. 50, Dec. 2009.

[35] R. Reboucas, J. Sauve, A. Moura, C. Bartolini, and D. Trastour, "A decision support tool to optimize scheduling of it changes," in *2007 10th IFIP/IEEE International Symposium on Integrated Network Management*, 2007, pp. 343–352. doi: 10.1109/INM.2007.374799.

[36] A. Mahalle, J. Yong, and X. Tao, "Itil processes to control operational risk in cloud architecture infrastructure for banking and financial services industry," in *2018 5th International Conference on Behavioral, Economic, and Socio-Cultural Computing (BESC)*, 2018, pp. 197–200. doi: 10.1109/BESC.2018.8697294.

[37] AXELOS, *ITIL4 Foundations*, 1st ed. Norwish: The Stationary Office, 2019.

[38] G. T. G. Neto, W. B. Santos, P. T. Endo, and R. A. Fagundes, "Multivocal literature reviews in software engineering: Preliminary findings from a tertiary study," in *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2019, pp. 1–6. doi: 10.1109/ESEM.2019.8870142.

[39] V. Garousi, M. Felderer, and M. V. Mäntylä, "Guidelines for including grey literature and conducting multivocal literature reviews in software engineering," *Information and Software Technology*, vol. 106, pp. 101–121, 2019, issn: 0950-5849. doi: https://doi.org/10.1016/j.infsof.2018.09.006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584918301939.

[40] A. Mair, *How to implement effective devops change management*, Feb. 2021. [Online]. Available: https://coralogix.com/blog/how-to-implement-effective-devops-change-management/.

[41] S. Pedersen, *Tips on using devops maturity assessments to manage change*, Jul. 2021. [Online]. Available: https://dzone.com/articles/tips-on-using-devops-maturity-assessment-to-manage.

[42] R. Lachhman, *Change management in the world of continuous delivery*, Sep. 2021. [Online]. Available: https://harness.io/blog/change-management-continuous-delivery/.

[43] J. Bloomberg, *Devops versus itil: How to win the battle over change management*, Jan. 2020. [Online]. Available: https://devops.com/devops-versus-itil-how-to-win-the-battle-over-change-management/.

[44] *Can devops tools support sustainable change management processes?* May 2018. [Online]. Available: https://www.newhorizons.com/article/can-devops-tools-support-sustainable-change-management-processes.

[45] *Devops change management in the enterprise world*, Aug. 2019. [Online]. Available: https://clearbridgemobile.com/devops-change-management-in-the-enterprise-world/.

[46] S. Zorah, *Bridging devops and itil*, Sep. 2015. [Online]. Available: https://www.atlassian.com/blog/devops/bridging-devops-itil.

[47] H. M. Johng, A. Kalia, J. Xiao, M. Vukovic, and L. Chung, "Harmonia: A continuous service monitoring framework using devops and service mesh in a complementary manner," in. Oct. 2019, pp. 151–168, isbn: 978-3-030-33701-8. doi: 10.1007/978-3-030-33702-5_12.

[48] G. Kim, *Trust me: The devops movement fits perfectly with itsm*, Mar. 2014. [Online]. Available: https://theitsmreview.com/2014/03/trust-devops-movement-fits-perfectly-itsm/.

[49] *Is the devops movement making change management obsolete?* Mar. 2016. [Online]. Available: https://www.ahead.com/resources/is-the-devops-movement-making-change-management-obsolete/.

[50] S. O'BRIEN, *Seven effective tips to get the most out of your devops*, Apr. 2020. [Online]. Available: https://devops.com/seven-effective-tips-to-get-the-most-out-of-your-devops/.

[51] L. Evenstad, "Delivering success with devops.," *Computer Weekly*, pp. 23–26, 2015, issn: 00104787. [Online]. Available: https://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=111934646&lang=pt-pt&site=ehost-live&scope=site.

[52] J. Boks, *Challenging the cumbersomeness of change management processes*, Sep. 2018. [Online]. Available: https://dzone.com/articles/challenging-the-cumbersomeness-of-change-management.

[53] ——, *Balancing change management, its complexity and agility*, Aug. 2018. [Online]. Available: https://dzone.com/articles/balancing-change-management-its-complexity-and-agi.

[54] *Devops process: Streamlining change approval*, Dec. 2021. [Online]. Available: https://cloud.google.com/architecture/devops/devops-process-streamlining-change-approval.

[55] *Does successful change management require devops? devops?* Mar. 2021. [Online]. Available: https://digital.ai/catalyst-blog/does-successful-change-management-require-devops.

[56] B. GOLDEN, *State of devops report 2020 focuses on the "how" of devops*, Jun. 2021. [Online]. Available: https://tanzu.vmware.com/content/blog/state-of-devops-report-2020-focuses-on-the-how-of-devops.

[57] B. Griffith, *The guide to building a devops change management plan*, Jun. 2019. [Online]. Available: https://victorops.com/blog/the-guide-to-building-a-devops-change-management-plan.

[58] *Manage change*, Jul. 2021. [Online]. Available: https://docs.microsoft.com/en-us/azure/devops/cross-service/manage-change?view=azure-devops.

[59] C. G. MCDOWELL, *If devops is required, then it's all about change management*, Apr. 2018. [Online]. Available: If%20DevOps%20is%20Required,%20Then%20It%E2%80%99s%20All%20About%20Change%20Management.

[60] C. Melendez, *How does devops handle change management?* Dec. 2017. [Online]. Available: https://dzone.com/articles/how-does-devops-handle-change-management.

[61] C. Riley, *Change your thinking about change management*, Aug. 2014. [Online]. Available: https://devops.com/change-thinking-change-management/.

[62] H. Topi and G. R. Spurrier, “Invited paper: A generalized, enterprise-level systems development process framework for systems analysis and design education,” *J. Inf. Syst. Educ.*, vol. 30, pp. 253–265, 2019.

[63] G. Thomas, *How to Do Your Case Study*. SAGE Publications, 2015, isbn: 9781473943612. [Online]. Available: https://books.google.pt/books?id=CMiICwAAQBAJ.

[64] R. K. Yin, *Case Study Research: Design and Methods (Applied Social Research Methods)*, Fourth Edition. Sage Publications, 2008, isbn: 1412960991.

[65] R. Jabbari, N. Ali, K. Petersen, and B. Tanveer, “What is devops?: A systematic mapping study on definitions and practices,” May 2016, pp. 1–11. doi: 10.1145/2962695.2962707.

[66] J. Faustino, D. Adriano, R. Amaro, R. Pereira, and M. M. da Silva, “Devops benefits: A systematic literature review,” *Software: Practice and Experience*, vol. 52, no. 9, pp. 1905–1926, 2022. doi: https://doi.org/10.1002/spe.3096. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/spe.3096. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.3096.

Appendixes

Appendix A

Interviewee	How would you describe it (DevOps) with your own words?
I_1	*"An independent team that can deliver value to the customer. Each team should have all required resources (human and tech) to deliver value to customer."*
I_2	*"It is a methodology that encompasses an autonomous team that can work in a small time delivery more software. Delivery value to the client as fast as possible and we are continuously improving. It is not, but on the next iteration we are improving and are constantly delivering small pieces of software with business value. It is having the same time to deliver software, but using it wisely and having tools and processes that ensure that agility. It is very process-driven: I visualize the DevOps lifecycle: build, test, release, deployment and then monitoring and feedback, change, plan, code, build, test and so on... This is the most valuable part, where the monitoring has the greatest impact on our planning because based on the feedback we can keep working (and adapt) our development continuously improving."*
I_3	*"An agile methodology that has to do with choosing a method of working in teams with different types of engineers, managers and using a methodology such as SCRUM, Kanban. A way to define how to deliver software quickly and agile."*
I_4	*"It is a set of tools and processes defined by a team and inherent infrastructure, in which the development is continuous, e.g. that streamlines the development test and delivery of a solution."*

I_5	*"It is everything (the basis) where we can work in a better way (gels developers and other coworkers), this way enabling software delivery."*
I_6	*"It's the set of best practices that allows us to build, ship, deploy, and maintain software in a more agile manner. By agile I mean, fast time to market, deploy more often and in an incremental way and with less friction between between developers and the other stakeholders of the system."*
I_7	*"It's an automated process in development in which since the beginning of the process (ticket creation, development and integration (with pipelines to aid in the testing and releases). In sum, all the processes to deliver software."*
I_8	*"It is related to all the operations related to software development."*
I_9	*"How is it possible to optimize the development process that the team is working on. For example, by using pair-design to optimize the work process, we can develop more collaboration and accelerate the learning and hand-off process."*
I_{10}	*"A combination of team development and operation together with some sort of best practices to increase the process of bringing changes to production more efficiently. "*

Appendix B

Interviewee	Can you describe its potential benefits?
I_1	*"It eliminates "blockers", as (in non-DevOps methodologies) we always need someone and needs to be validated. (DevOps) eliminates dependencies. The way we are working is that it can be testable and whether we can deliver or not. It's having a more consistent work experience. And we know what we're waiting for and we're not constantly thinking 'Oh, this is new! What should we do?'"*
I_2	*"The customer is not waiting for an entire quarter for a feature/piece of software and by delivering in small bits, the client can participate in the development loop and adapt the development to tailor the customer needs. We always have customer feedback, which can improve our product to the due to the constant feedback."*
I_3	*"It gives a sense of having significat teamwork, speed up the delivery of software. Serve as a guideline for where to turn in terms of methodologies, frameworks and guide the development process. Having a place where the process is defined."*
I_4	*"I see it as a normalization of processes, less time wasted on repetitive tasks (human) and less manual tasks (such as run deploys, tests etc). By having automated processes and having less human-prone errors."*
I_5	*"It improves the developer experience, in a way that we do not have so many problems related to the development and we can focus more on our tasks. The focus for developers is on software development and not on operations (environments, pipelines, test execution)."*
I_6	*"It reduces costs to deploy and maintain the system; enables delivery of small changes more often, incremental/continuous development and deployment; brings value to the final client quicker; reduces friction of the feedback loop between the system end users (and other stakeholders) and the system developers; break communication silos within the company that tend to be translated to the system final design."*

I_7	*"It normalizes the development, better reliability of development (end result), automated process. Regarding the organization, it is ensured by automation, because it helps not to have so many errors, leading to a deterministic developer experience."*
I_8	*"It improves communication and promotes faster software delivery."*
I_9	*"It improves and optimizes the development process."*
I_{10}	*"It improves communication, allows fast delivery cycles, better alignment."*

Appendix C

Interviewee	Can you justify why (would it be easy/neutral/hard to implement these practices)?
I_1	*"**Application Monitoring** and **Automated Dashboards** are quite easy to configure. Based on HTTP and body responses it becomes very simplistic whether everything is okay or not, in a very linear way. So the only thing we need to configure is a tool that is looking at the traffic and seeing the requests and the responses, and the latencies of the responses. So it allows us to have a lot of information from the server side.* ***Continuous Delivery**, **Automated Deployment**, **Continuous Deployment** are hard, in order to have it fully automated there are a lot of gears that need to work smoothly for it to work. If one fails, the whole process fails. You must have a set of tools/tools/processes. When it does 'Nice job!', when not 'Where did it fail? Why did it fail?'. It is a 'black-box' (tools and infrastructure around it to ensure the continuity of development) and when there are failures it is difficult to inspect. So, the implementation can be very difficult.* ***Continuous Testing** is very hard because running all the tests when there is a code change is not feasible, in practice the best thing was to have a way to run only the tests whose line of code was changed and test at most the module where the change was introduced. When we have Continuous Testing it is mandatory not to have all tests running when we change a line of code. Traceability is extremely difficult."*

I_2	*"**Continuous Planning** is easy as it is something we do on a daily basis, but we must at least have a mature team and know the product. If we have a team that is constantly changing, the entire team may not know the entire product (software), so planning tends to be harder. Still, I find it easy, as we need to have a mature team in DevOps.* ***Feedback Loops between Dev and Ops** seems to be easy if the team is mature enough.* ***Continuous Testing** is hard because it is not only about adding more tests, and having irrelevant tests may not be the best approach.* ***Process Standardization** is easy because it is a matter of management alignment.* ***Infrastructure as Code (IaC)** is very hard because it is something very complex to manage.* ***Stakeholder Participation** is hard because it depends on the customer participation level and on the team maturity."*
I_3	*"**Continuous Planning**, **Feedback Loops between Dev and Ops** and **Stakeholder Participation** are very easy because it has a more human component. There are always problems with people's calendar conflicts, but to solve it, just book and organize that you can solve with communication.* ***Stakeholder Participation**, **Continuous Deployment**, **Automated Deployment** and **Continuous Testing** are hard not because of the implementation, but all the friction in the process during development while the various teams. When there is automation or technical effort because in my experience as a software engineer these are hard."*

I_4	*"**Continuous Monitoring**, **Application Monitoring** and **Automated Dashboards** are easy because there are tools ready to implement it.* ***Continuous Integration** is hard because we need pipelines, e.g. technical effort to implement them.* ***Continuous Deployment** is hard due to the technical efforts necessary to ensure the automated process. It becomes easier when Automated Deployment is already implemented.* ***Continuous Delivery** and **Automated Deployment** are hard due to technical effort to build the pipelines, similarly to Continuous Deployment.* ***Automated Testing** is hard to implement, as well as introducing it within the process and due to technical effort to build the pipelines."*
I_5	*"**Continuous Planning** is harder than it seems. It is an iterative and learning process. How to communicate to our stakeholders what we are going to work on. Learn about the error so we can have more robust planning.* ***Feedback Loops Between Dev and Ops** easy, because the hardest part is the personal level on how to receive criticism.* ***Continuous Monitoring**, **Measure Performance Metrics (CI, Test & Ops)** and **Application Monitoring** are hard because developing from scratch and choosing how to do it is complex.* ***Continuous Integration** and **Continuous Delivery** is easy because we are always integrating new things to our code base, so it seems easy on that note.* ***Continuous Testing** and **Automated Testing** is easy if taken from the very beginning: it is easy to maintain because it is part of the development.* ***Stakeholder Participation** is hard even being part of communication, it is not straightforward. If the stack is too complex, we may not know with whom to talk, who are the stakeholders. Also, how we communicate something among the many stakeholders."*

I_6	*"**Continuous Planning**, **Feedback Loops between Dev and Ops** and **Stakeholder Participation** are easy because they can be incorporated into the team's weekly activities.* ***Continuous Monitoring** is hard, because it requires building infrastructure to produce, collect and expose that data. It also requires a team, or some sort of effort, to actively check and act upon that monitoring.* ***Measure Performance Metrics (CI, Test & Ops)** performance engineering is always very hard, as it requires in-depth systems design knowledge (e.g. latencies that each component might introduce in the system), requires deep knowledge on how the technology used to build the product works under the hood (e. g. garbage collector), requires building a systematic approach to conduct deterministic and reproducible experiments, require some mathematical statistical knowledge to be able to properly analyze the results (e.g. understand percentiles, statistical noise and outliers)* ***Application Monitoring** is hard, because it needs to know what to measure and how to measure. The second part might be easy or not depending on the technology being used.* ***Automated Dashboards** might be hard as it requires making sure all the edge cases are covered.* ***Prototyping Application** should always be easy to come up with a very draft version of any system or feature.* ***Continuous Deployment** is hard, because deployment typically entails a lot of steps which might be very error prone and lead to malformed deployment.* ***Automated Deployment** is hard, because automation might be hard as it requires to make sure all the edge cases are covered, even more to deploy and bootstrap fully functional, client-ready systems.* ***Continuous Delivery** is hard, because delivery typically entails a lot of steps which might be very error prone and lead to malformed deployment.* ***Continuous Integration**, **Continuous Testing**, **Automated Testing** and **Infrastructure as Code (IaC)** are easy, because it is a well-known practice with a lot of ready-to-use resources on how to implement."*

I_7	*"**Continuous Planning** is easy because it demands only communication, the requirements and discussion with stakeholders (which take part of the planning). It solely depends on communication, which makes it easy.* ***Feedback Loops between Dev and Ops** is easy because it depends again on communication, so we can improve anything that needs to be improved in the team, from technical to people.* ***Continuous Monitoring** and **Measure Performance Metrics (CI, Test & Ops)** is easy because we already have technologies that aid us in this task* ***Continuous Integration** is easy because we have tools that are mostly out-of-the-box, so implementing this process seems to be easy to implement.* ***Prototyping Application** is easy because it is only a prototype of a feature, so it is simple to implement.* ***Continuous Deployment**, **Automated Deployment** and **Continuous Delivery** is hard because it depends on the scenario for each process. There are more nuances depending on the product we are working on.* ***Continuous Testing** and **Automated Testing** are easy because they can be ensured by integration in the pipeline, and also during the SDLC.* ***Stakeholder Participation** is easy because they just need to be available and again communicate."*
I_8	*"**Continuous Planning** seems to be hard because it depends on the context of the company, this context is related to the company's constantly changing vision and it is hard to continuously plan when the company's vision keeps changing.* ***Feedback Loops between Dev and Ops** is easy because it is a matter of dedicated time to do it properly or create processes that help it, but the process itself seems easy.* ***Process Standardization** and **Stakeholder Participation** are easy because it is a matter of dedicate time to properly."*

I_9	*"**Continuous Monitoring** and **Automated Dashboards** might be easy because such tools are easy to configure and might have been implemented previously or in conjunction with other automation processes.* ***Measure Performance Metrics (CI, Test & Ops)** is hard because to have such metrics we need time to have something traceable and have realistic outcomes. We need at least 6 months to have relevant data. Maybe not the implementation, but the time to have such data.* ***Automated Deployment** I find it hard because, in comparison to manual deployments that depend on us (the team), it can be dangerous because we might have something that is not ready (quality-wise).* ***Continuous Delivery** and **Continuous Testing** I would say easy if we already have implemented tools and some previous automation.* ***Process Standardization** I find it hard because it must be something very well defined and have vision, as the process must evolve, but still we have to be certain about what we are aiming for* ***Stakeholder Participation** I think it is hard that depends on the stakeholder openness to be in this implementation, as there might have communication issues (some people are easier to talk to) but it can be hard as you have to have the right stakeholder, because some people are key when developing a new feature."*

I_{10}	*"**Continuous Planning** is hard because it needs to be based on a common foundation and if that changes all the time, it is tricky to align with everybody. People from different levels have different ideas and if that is not given and no common structure is in plan, it is hard to do continuous planning, because it has to be aligned with stakeholders, marketing and sales. If the foundations/setups are prioritized correctly, Continuous Planning is easy.* ***Application Monitoring**, **Continuous Monitoring**, **Measure Performance Metrics (CI, Test & Ops)** and **Automated Dashboards** are easy, because we just need to know what we want to monitor and set priorities of what needs to be monitored.* ***Continuous Integration** is hard and depends a lot on the setup (of the business).* ***Prototyping Application** is easy, because it is only an early MVP (minimum viable product) and it is beneficial from a product perspective.* ***Continuous Deployment** and **Automated Deployment** are hard, as it depends on your business, because you have to coordinate with the users, have training, release notes* ***Continuous Delivery** is hard, in order to maintain the system working (backwards compatibility), how to set the processes in place so the end user is not affected* ***Process Standardization** is hard, because you have to find the right level of standards and not make it too complex, it still needs to be simple* ***Stakeholder Participation** is easy, because it is about finding a way to share and inform your work with somebody in a way they can understand it (effective communication)."*

Appendix D

Practices	Change Management Stage (Standard Changes)								
	Create RFC	**Record RFC**	**Review RFC**	**Access and Evaluate Change**	**Authorize Change and Build Test**	**Coordinate Change and Build Test**	**Authorize Change Deployment**	**Coordinate Change Deployment**	**Review and Close Record**
Continuous Planning	3	1	3	8	1	2	-	3	1
Feedback Loops Between Dev and Ops	1	1	1	2	-	1	-	3	2
Continuous Monitoring	4	3	3	3	-	-	-	-	4
Measure Performance Metrics (CI, Test and Ops)	3	2	2	1	-	-	-	-	3
Automated Feedback for Performance Models and Predictions	1	1	1	1	1	1	1	1	1
Application Monitoring	3	3	3	1	-	-	-	-	1
Automated Dashboards	3	3	2	2	2	1	2	1	2
Continuous Integration	5	4	3	3	5	5	4	4	1
Prototyping Application	1	1	2	3	1	1	-	-	-

Continuous Deployment	4	3	3	3	3	4	5	7	3
Automated Deployment	4	3	3	3	3	4	5	8	3
Continuous Delivery	4	3	5	4	4	5	4	6	5
Continuous Testing	3	2	6	3	3	5	1	2	2
Automated Testing	2	2	5	2	4	5	1	2	1
Process Standardization	4	3	5	3	2	4	2	3	4
Infrastructure as Code (IaC)	2	1	3	2	1	2	1	3	1
Stakeholder Participation	5	2	3	2	-	1	1	2	4

Practices	**Change Management Stage (Emergency Changes)**						
	Review RFC	**Access and Evaluate Change**	**Authorize Change and Build Test**	**Coordinate Change and Build Test**	**Authorize Change Deployment**	**Coordinate Change Deployment**	**Review and Close Record**
Continuous Planning	4	6	3	4	2	3	3
Feedback Loops Between Dev and Ops	1	1	-	2	-	3	2
Continuous Monitoring	2	3	3	1	-	2	3
Measure Performance Metrics (CI, Test and Ops)	1	3	3	2	-	2	4

Automated Feedback for Performance Models and Predictions	-	1	1	1	-	1	3
Application Monitoring	2	3	2	1	-	2	4
Automated Dashboards	3	2	1	-	1	1	3
Continuous Integration	3	4	5	5	3	4	2
Prototyping Application	2	2	1	3	-	-	-
Continuous Deployment	3	4	3	4	5	6	3
Automated Deployment	3	4	3	4	4	7	3
Continuous Delivery	4	5	4	4	3	5	3
Continuous Testing	1	3	4	5	-	2	-
Automated Testing	1	4	6	6	1	3	1
Process Standardization	4	3	3	3	2	2	2
Infrastructure as Code (IaC)	2	1	2	2	1	2	1
Stakeholder Participation	5	3	3	4	2	3	5

Practices	Change Management Stage (Normal Changes)						
	Review RFC	**Access and Evaluate Change**	**Authorize Change and Build Test**	**Coordinate Change and Build Test**	**Authorize Change Deployment**	**Coordinate Change Deployment**	**Review and Close Record**
Continuous Planning	5	5	4	3	1	5	3
Feedback Loops Between Dev and Ops	2	2	1	1	-	3	3
Continuous Monitoring	3	5	2	1	1	2	5
Measure Performance Metrics (CI, Test and Ops)	1	5	2	2	1	2	5
Automated Feedback for Performance Models and Predictions	-	1	1	1	-	1	3
Application Monitoring	2	4	1	1	1	2	4
Automated Dashboards	2	2	1	-	2	1	4
Continuous Integration	2	4	6	5	3	4	2
Prototyping Application	-	2	2	1	2	-	-
Continuous Deployment	2	4	3	4	4	6	2
Automated Deployment	1	3	2	3	3	6	1
Continuous Delivery	5	6	5	5	4	6	4
Continuous Testing	2	5	5	5	1	3	3

Automated Testing	3	5	5	5	1	3	3
Process Standardization	5	4	2	4	1	2	4
Infrastructure as Code (IaC)	-	1	1	-	-	1	-
Stakeholder Participation	5	5	2	5	-	2	6

Appendix E

Interviewee	Could you provide why and how these practices can be related (for a Standard Change)?
I_1	*" **Continuous Integration**, **Continuous Deployment**, **Automated Deployment**, **Continuous Delivery**, **Continuous Testing** and **Process Standardization** - Everything that is a procedural change must be evaluated and monitored and implemented consciously and with a very defined, hence the formality of all processes. And so that it can be properly documented for those who come to know or be able to know what the process was that we were supposed to be defining."*
I_2	*" **Continuous Planning** I think it is related to (access and evaluate change) because we need to review what was created and assess if it is value, also we need to think how this change will evolve: what is the purpose, who is the target of such change.* ***Stakeholder Participation** because it is an important component in the begin and and the end of the process.* ***Continuous Monitoring** because based on the monitoring we can know if that change is necessary, e.g. work for a general purpose instead of addressing just one issue. **Continuous Integration** because through automation we can have the while process of developing software.* ***Continuous Testing** because you have to verify what exists we just need to have an authorization, we just need to coordinate what needs to be deployed. You actually do not need something to be authorize, it just needs to be coordinated and not authorized.* ***Automated Testing** because it is something new we just need to communicate to ensure we have good automated tests."*

I_3	*" **Continuous Delivery**, **Continuous Deployment**, **Automated Deployment**, **Automated Testing** and **Continuous Integration** because it cannot change the functioning without being evaluated, accepted and reviewed. Regarding the pipeline, they need to guarantee the entry of my change will not be harmful to the system. The whole process of testing pipelines and how things are working and there's a whole process where this change will be integrated that is important to be executed."*
I_4	*" **Continuous Monitoring**, **Application Monitoring** and **Automated Dashboards** it is needed to be created and evaluate if the change is actually needed. If you want to change this systems only creation and evaluation is necessary.* ***Continuous Integration** because changes in these processes are extremely important and should be validated thoroughly.* ***Continuous Deployment**, **Automated Deployment** and **Continuous Delivery** one of the most important process where it was more bureaucratic, e.g. for each new deploy was extremely bureaucratic and had a set of requests and documents to be done, tests after deploy, rollback mechanisms and had to be validated."*
I_5	*" **Continuous Planning** evaluate change because it is when we can verify if it is going to bring value to our product or not.* ***Feedback Loop Between Dev and Ops** create tickets about the retro sessions and improve the product.* ***Continuous Monitoring** create tickets based on what you want to monitor.* ***Measure Performance Metrics** create tickets based on what you want to monitor* ***Continuous Deployment**, **Automated Deployment** and **Continuous Delivery** only coordinate because it is the only* ***Stakeholder Participation** coordinate the communication between all team members."*

I_6	*" **Continuous Planning** as we are continuously planning, we evaluate why the change is necessary and coordinate when it should be deployed.* ***Feedback Loop Between Dev and Ops** evaluate the changes and coordinate how they are deployed.* ***Continuous Monitoring** and **Measure Performance Metrics** authorize changes that were implemented in the past and close.* ***Continuous Integration** review if it can be done or not is necessary.* ***Prototyping Application** evaluate only if it is worth it is necessary.* ***Continuous Deployment** review if it can be done or not is necessary.* ***Automated Deployment** review if it can be done or not is necessary.* ***Continuous Delivery** review if it can be done or not is necessary.* ***Continuous Testing** review if it can be done or not is necessary.* ***Automated Testing** review if it can be done or not is necessary.* ***Process Standardization** create and then close is necessary."*

I_7	*" **Continuous Planning** create/record/review because we need to have the change and evaluate the impact and also what's next in the development.* ***Feedback Loops Between Dev and Ops** review and evaluate to understand what needs to be addressed ans close because we get to finish the task (discussion).* ***Continuous Monitoring** and **Measure Performance Metrics** important because based on the data we can assess the change, based on the status of the application.* ***Continuous Integration**, **Continuous Deployment**, **Automated Deployment** and **Continuous Delivery** a change may need authorization and coordinate this change to be deployed.* ***Prototyping Application** review the ticket as based on the results of the software iteration, we can also assess.* ***Stakeholder Participation** creation and authorization of the deployment, also closing the change."*
I_8	*" **Continuous Planning** it is import to have a frequent review to re-evaluate and do adjust, and also coordinate (communicate) this change.* ***Stakeholder Participation** it is important to have them created and communicated. Sometimes it seems too informal and needs to be more effectively communicated."*

I_9	*" **Continuous Planning** because it it is something that can be done on our side as a team, so we have autonomy on our decisions as a team.* ***Continuous Monitoring** and **Measure Performance Metrics** because I can work on coordination but not something that can be manually, but automatically defined, although it has to be coordinated with the rest of the team.* ***Continuous Delivery** because it it is about delivering so we need to review it, as we are releasing software. That is review and coordinate such change.* ***Process Standardization** because we are involved in these to make sure what we are doing will improve the process.* ***Stakeholder Participation** review and close and review so we can optimize communication."*

I_{10}	*" **Continuous Planning** because you need to have new information and based on that, evaluate, authorizing and coordinate that with the team.* ***Feedback Loops Between Dev and Ops** just the end of the process.* ***Continuous Monitoring** and Measure Performance just the end of performance.* ***Application Monitoring** not creating but checking what was recorded.* ***Continuous Integration** Authorization and coordination phase.* ***Prototyping Application** from creating the change until the coordination of this change.* ***Continuous Deployment** and **Automated Deployment** from coordinating the change coordinate its deployment. We must have some standards pin lace in order to set it up and works.* ***Continuous Delivery** you need to understand who are he sponsors, the goals and finally to coordinate how it is delivered.* ***Continuous Testing** and **Automated Testing** need to get the requirement, tested accordingly and then coordinates its change deployment.* ***Process Standardization** everything is a process in place and must be taken into consideration.* ***Stakeholder Participation** creation and coordination, not the authorization because it already happens in the beginning (the feedback is from feedback mostly)."*

Interviewee	**Could you provide why and how these practices can be related (for an Emergency Change)?**
I_1	*" My answer will be the same as before (regarding the practices for standard changes). The difference here are how we prioritize what need to be changes because we are in an emergency. Still, the process is going to be followed."*

I_2	*" Everything that is more critical have a greater impact, so it is different how we approach such changes. As the priority changes and you will have more people to look into. For these changes, we need to speed up things (change priority) and have someone to authorize it, if that is the case. So based on that:* ***Continuous Planning*** *because when looking at the board, we can tell what it more prioritized and should be tackled right away. We just have to plan differently.* ***Continuous Deployment*** *and* ***Continuous Delivery*** *because we do need authorization (only if the upper level is too much involved) for something that is urgent and needs to be escalated.* ***Stakeholder Participation*** *they need to be aware of what that change entails."*
I_3	*" It should be the same (as it is for standard changes), because we do not want to introduce any harm to the system."*
I_4	*"* ***Continuous Monitoring***, ***Application Monitoring***, ***Measure Performance Metrics***, ***Automated Dashboards*** *extremely important to create and coordinate the change. To validate the changes of emergency because they are tools that ensure that the system is recovering or not.* ***Continuous Integration***, ***Continuous Deployments***, ***Automated Deployment***, ***Continuous Delivery*** *the deliverable also guarantee that will ensure the change was effective and in an emergency it must be validated thoroughly and all steps are required."*
I_5	*"* ***Continuous Planning*** *even being an emergency, we need to ensure quality and not have the problem again as we want to ensure confidence on the part of our end users (clients) and also by escalating the priority. Detailed attention to avoid errors.* ***Continuous Testing*** *and* ***Automated Testing*** *even being an emergency we need some confidence level in the change being performed and detect regressions (even if is not an emergency)* ***Stakeholder Participation*** *it is important to have the stakeholder's participation in each part of the process, specially when there is an emergency."*

I_6	*" We have to ensure the same process considering how critical the execution is, i.e. priority changing. So I am answering the same as previously."*
I_7	*" I think I will choose the same practices and reasons with the only difference by adding Authorize Change and Build Test step as part of the CM process (except for Continuous Planning, Feedback Loops and Stakeholder Participation)"*
I_8	*" The practices (and change management phases) are the same in my opinion, there is only a change in priority to be considered."*
I_9	*" Besides what I answered on the previous question, I would add* ***Continuous Planning*** *because this practice we need to be involved in every step in the change management process when we have an emergency."*
I_{10}	*" I will answer the same as the previous question, but will only add the "Authorization" and "Coordination" are not needed because it is an emergency and it can be skipped."*

Interviewee	**Could you provide why and how these practices can be related (for a Normal Change)?**
I_1	*" My answer will be the same as before (regarding the practices for standard and emergency changes). The difference here are how we prioritize what need to be changes because we are in an emergency. Still, the process is going to be followed."*

I_2	*" As I consider this something easier, I think the same practices can be the same as the emergency ones but without the priority and the authorization chain.* ***Continuous Planning*** *because we can see a board and can do something as a team, we can define our own priority as a an autonomous team.* ***Continuous Deployment*** *and* ***Continuous Delivery*** *because we do not need authorizations for something that is not urgent and does not to be escalated.* ***Stakeholder Participation*** *I think there is more participation of the normal changes, as we do not have something urgent to be fixed."*
I_3	*" It should be the same (as for standard changes and emergency changes), because we do not want to introduce any harm to the system."*
I_4	*"* ***Continuous Monitoring****,* ***Measure Performance Metrics****,* ***Application Monitoring****,* ***Automated Dashboards*** *The tests are not necessary otherwise they would be not efficient/effective, we would be "testing the tests". We need to ensure we create and deploy changes to these tools.* ***Continuous Integration****, Continuous Deployment, Automated Deployment and Continuous Delivery even being normal changes, they need to go though the entire process as it has the most delicate parts of the SLDC an we need to ensure quality upon deploy.* ***Continuous Testing*** *and* ***Automated Testing*** *only for scenarios where we need to cover it with tests besides reviewing this change and closing it."*
I_5	*" The same as the previous changes (standard and emergency ones). The most important aspect is to guarantee quality that is achieved by the automated process that are going to be perform all stages until deployment."*
I_6	*" We have to ensure the same process considering how critical the execution is, i.e. priority changing."*

I_7	*" What change it the task prioritization, not the process per se, so I would say I will stick to the same (change management) phases."*
I_8	*" **Continuous Planning** as it is already part of the plan, we can review it and then coordinate when it will be deployed and ensure it is aligned with the product planning.* ***Feedback Loop Between Dev and Ops** it allows things to advance faster and it is key (all except authorize). As it is something it not necessary authorization, just to analyzer and coordinate.* ***Process Standardization** and **Stakeholder Participation** it allows that normal changes will not have an impact on the planning, if we follow the process we will not have disturbances on development, then we will have continuous development and will not affect our planning. Not necessary on authorization but coordination."*
I_9	*" **Continuous Planning** because this practice we need to be involved in every process when we have a normal change.* ***Continuous Testing** because we can review the request (for change) and no need for a previous review.* ***Process Standardization** because I can review the request and evaluate such change."*
I_{10}	*" I do not see much difference to a standard change, probably the priority is different and being part of something already planned. So I will choose the same."*

Appendix F

Interviewee	How do you think DevOps is helpful when there is a change?
I_1	*"An independent change is more achievable as it's easier to change a "small" team than to change the whole company. Each team can choose which processes are best for them (following base standards). The context given to the team that is autonomous in assuming and integrating this change. As there are no such dependencies with third parties, it is much easier to integrate these changes into this project."*
I_2	*"DevOps helps us to have a lean planning, deliver faster, get feedback gets sooner (due to the monitoring tools and also stakeholder participation) so we can improve what we are delivering for each change we are faced. This way, we would never have the product not aligned to what the clients was aiming for, in comparison to delivering software as a whole product."*
I_3	*"I think so because due to the tools and frameworks that are the basis of this development process, it creates a foundation of trust when we make a change and that the project will continue to work and be delivered."*
I_4	*"It ensures we have all the tools in place, that this change is safer to be delivered. We will know the impact in all phases in the life cycle of the change (development, tests, deployment and integration). We know this change will be implemented and validated and also guarantee that this process is automated and by being so, it is a big advantage."*
I_5	*"It is important because it helps to automate processes such as Authorization Change Deployment and makes us have changes delivered faster to our end-users. Removes the manual/human side of the entire process especially for deployment purposes."*
I_6	*"DevOps helps: Automate the process to apply/deploy the change; monitor the impact and KPIs of the change; enables the communication among all the stakeholders that might be affected by the change; automate the reversion of the change (if needed; reduce the overall cost to apply and maintain that change"*

I_7	*"DevOps is helpful to streamline the SDLC and this way a change is traceable, as we have process standardization, even if we have changes, as we have processes and automation, the change is easily adopted when using a DevOps approach."*
I_8	*"It has to do with two things: standardization and communication loop are essential. Standardization helps us to react better to changes, as we know how to react technically, but by having a process defining how to react helps a lot in the initial process in the cognitive load when processing this change. On the communication side, when we make a change where we have the stakeholders involved, we are ensured the change is known by all stakeholders and makes sense in different contexts, such as user experience and product end user. Automation is also important to ensure all things are delivered. Continuous Integration is crucial too, as if we have a process too bureaucratic and complex, our delivery will be slowed down. By having a continuous approach it is much easier to introduce that change. Not having a standardized process and automation, this change will take longer and will be harder as we do not have it within the development life cycle, also how we face such change and the effort taken to be considered."*
I_9	*"I think it is how we can improve the process and by having DevOps we are more prepared for emergencies. (In Design) I know that the development team is also ready if we have something more urgent, and it is great in the sense of deliverables that we can achieve due to the collaborative environment."*
I_{10}	*"It is helpful to communicate, collaborate, sync, align and also react quickly to situations."*