Adopting DevOps in the Real World: A Theory, a Model, and a Case Study

Article *in* Journal of Systems and Software · July 2019

DOI: 10.1016/j.jss.2019.07.083

Welder Luz
University of Brasília
6 PUBLICATIONS **315** CITATIONS
SEE PROFILE

Gustavo Pinto
Federal University of Para
159 PUBLICATIONS **3,744** CITATIONS
SEE PROFILE

Adopting DevOps in the Real World: A Theory, a Model, and a Case Study

Welder Pinheiro Luz[a], Gustavo Pinto[b,*], Rodrigo Bonifácio[c]

[a]*Brazilian Federal Court of Accounts*
[b]*Federal University of Pará*
[c]*University of Brasília*

Abstract

DevOps is a set of practices and cultural values that aims to reduce the barriers between development and operations teams. Due to its increasing interest and imprecise definitions, existing research works have tried to characterize DevOps.

Nevertheless, little is known about the *practitioners' understanding* about successful paths for DevOps adoption. Therefore, our goal is to detail real scenarios of DevOps adoption, presenting a theory, a model, and a case study.

We used classic Grounded Theory to build a theory about 15 scenarios of successful DevOps adoption in companies from different domains across 5 countries. We proposed a model (i.e., a workflow for DevOps adoption) and evaluated it through a case study at a Brazilian Government institution. We used a focus group to collect the company perceptions about DevOps adoption, including the model' application.

This paper increments the existing view of DevOps by detailing real scenarios and explaining the role of each category during DevOps adoption. We provide evidence that *collaboration* is the core DevOps concern, contrasting with an existing wisdom that automation and tooling can be enough to achieve DevOps.

Altogether, our results contribute to: generating an adequate understanding of DevOps, from the practitioners' perspective; and assisting other institutions in the migration path towards DevOps adoption.

Keywords: DevOps, Grounded Theory, Software Development, Software Operations, Focus Group

1. Introduction

DevOps is a set of practices and cultural values that has emerged in the software development industry [1, 2, 3, 4]. Even before the existence of the term — a mix of "development"

[*]I am corresponding author

Email addresses: `welder.luz@tcu.gov.br` (Welder Pinheiro Luz), `gpinto@ufpa.br` (Gustavo Pinto), `rbonifacio@cic.unb.br` (Rodrigo Bonifácio)

and "operations" words [5] — companies like Flickr [6] had already pointed out the need to break the existing separation between the operations and software development teams. Since then, the term that although appeared without a clear delimitation, gained strengths and interests from companies that perceived the benefits of applying agile practices in *operation tasks*. DevOps claimed benefits include increased organizational IT performance and productivity, cost reduction in software lifecycle, improvement in operational efficacy and efficiency, better quality of software products, and greater business alignment between development and operations teams [4, 7, 8]. However, the adoption of DevOps is still a challenging task. Even though there is a plethora of information, practices, and tools related to DevOps, it is still unclear how one could leverage such rich, yet scattered, information in an organized and structured way to properly adopt DevOps.

Existing research works have proposed a number of DevOps characterizations, for instance, as a set of concepts with related practices [1, 2, 3, 4, 9, 10]. Although some of these studies leverage qualitative approaches to gather practitioners' perception (for instance, conducting interviews with them), they focus on characterizing DevOps, instead of providing recommendations to assist on DevOps adoption. Consequently, our **research problem** is that the obtained DevOps characterization provides a comprehensive understanding of the elements that constitute DevOps, but do not provide detailed guidance to support newcomers interested in adopting DevOps. As a consequence, many practical and timely questions still remain open, for instance: (1) Is there any recommended path to adopt DevOps? (2) Since DevOps is composed by multiple elements [2], do these elements have the same relevance, when adopting DevOps? (3) What is the role played by elements such as measurement, sharing, and automation in a DevOps adoption? To answer these questions, we need a holistic understanding of the paths followed in successful DevOps adoptions.

This paper is a continuation of a previous study [11]. In our previous study, we introduced a model (i.e., a workflow for DevOps adoption) based on the perceptions of practitioners from 15 companies across five countries that successfully adopted DevOps. The model was constructed based on a classic Grounded Theory (GT) approach, and makes clear that practitioners interested in adopting DevOps should focus on building a ***collaborative culture***, which prevents common pitfalls related to focusing on tooling or automation [12]. We instantiated our model in the Brazilian Federal Court of Accounts (hereafter TCU),[1] a Brazilian Federal Government institution. TCU was bogged down in implanting specific DevOps tools, repeating the same non-DevOps problems, with conflicts between development and operations teams about how to divide the responsibilities related to different facets in the intersection between software development and software provisioning.

When instantiated, our model helped TCU to change its focus to improve the collaboration between teams, and to use the tooling to support (rather than being the goal of) the entire process. In our previous work we briefly introduced this experience at TCU. Complementing this initial research, in this paper we report the current status of the DevOps adoption at TCU as a whole, including an empirical assessment of our model. We collect the TCU perceptions through a focus group with four of its directly involved professionals,

[1]http://www.tcu.gov.br/

two of each team (development and operations). Based on the results of the focus group, we review our theory in this paper, which is detailed using a well-known framework for building theories on software engineering [13]. The main contributions of this paper are the following:

- A review and extended description of our work that builds a model and a theory about DevOps adoption. This theory might support practitioners interested in adopting DevOps, based on evidence acquired from industry peers.
- An instantiation of this theory in a real world, non-trivial context. Specifically, TCU, as a government institution, is rather different from typical *tech companies* that have successfully reported the adoption of DevOps, which substantiate the DevOps potential, even in more traditional companies.
- The results of a focus group that evidenced that the use of our theory in TCU brought several benefits and now DevOps practices have been disseminated at TCU. We also report initial benefits of DevOps adoption in this company, in particular the introduction of an agile approach for software deployment and provisioning.

The rest of the work is organized as follows: Section 2 presents our method and the settings used to conduct this research. Section 3 describes our preliminary findings on what constitute DevOps, along with the main enablers and outcomes categories. In Section 4 we present our theory regarding a successful DevOps adoption, while Section 5 describes our three step model that one could use to adopt DevOps. Next, in Section 6 we discuss the findings of applying our model in a real world setting. Section 7 discusses some threats to validity, while Section 8 presents the related work. Finally, Section 9 concludes our work and suggest eventual research opportunity for future work.

2. Method and Settings

The general **goal** of this research is two fold. Fist, we aim to develop a model on DevOps adoption, considering the perspective of practitioners who contributed to the adoption of DevOps in the organizations they work for. Second we investigate the relevance of this model in a real DevOps adoption scenario. Tables 1 and 2 summarize our research goals using a GQM (Goals/Questions/Metrics) template [14]. We investigate two main **research questions**, which we further develop during our research. These main research questions are

(RQ1) How do practitioners characterize a successfully path for DevOps adoption? This is a broad question that motivates us to build a theory and a model for DevOps adoption. As discussed in this section, we further refine this general research question in a number of sub-questions, using a typical Grounded Theory approach (Section 2.1).

(RQ2) How does our model contribute to the adoption of DevOps on a specific scenario? Answering this question allows us to understand the relevance of our model, showing its practical implications in a real settings. We investigate this question using the focus group method (Section 2.2).

Analyze	**Object under measurement**
For the purpose of	*Understand the process of DevOps adoption.*
With respect to	*the objectives that motivate a DevOps adoption process as well as the concerns that might enable DevOps adoption or the concerns that correspond either to the benefits or to challenges related to DevOps adoption.*
From the view point of	*practitioners that have contributed to a previous effort on DevOps adoption.*
In the context of	*companies in different domains that have adopted DevOps.*

Table 1: GQM related to our goal of *characterizing DevOps adoption*

Analyze	**Object under measurement**
For the purpose of	*Understand the relevance of our DevOps adoption model.*
With respect to	*the guidance it provides on the activities that might lift the results of a DevOps adoption effort.*
From the view point of	*practitioners that are participating on a DevOps adoption effort.*
In the context of	*Brazilian Government Institution.*

Table 2: GQM related to our goal of *assessing the DevOps adoption model*

We address these questions using a qualitative research, and therefore we do not use any specific software engineering metric. Nonetheless, to investigate some of the benefits of adopting DevOps at TCU, we explore two metrics: the maximum number of daily and weekly deployments for a set of systems. Besides other benefits, deployment agility is a crytical achievement from adopting DevOps at TCU. Before that, deployments activities for these systems occur once a week,

We use a multi-method approach to achieve our results. First, we use Grounded Theory (GT) as the research method to build an explanation about how DevOps was successfully adopted in companies that claims to have made it. Based on this explanation, we proposed a model to guide new adopters. Second, we use the Focus Group method to explore in details a real experience on DevOps adoption, including the application of our model.

2.1. Grounded Theory

Grounded Theory was originally proposed by Glaser and Strauss [15]. As distinguishing features, GT has (1) the absence of clear research hypothesis upfront and (2) limited exposure to the literature at the beginning of the research. That is, GT is a theory-development approach (the hypothesis emerge as a result of a investigation), in contrast with more traditional theory-testing approaches [16]—e.g., those that use statistical methods to either confirm or refute pre-established hypothesis.

The motivation to use GT is due to three main reasons. First, GT is a consolidated method in other areas of research — notably medical sociology [17], nursing [18], education [19], and management [20]. More recently, GT is also being increasingly employed to study software engineering research topics [21, 22, 23]. Second, GT is considered an adequate approach to answer research questions that aims to characterize scenarios under a personal perspective of those engaged in a discipline or activity [18], which is exactly the scenario here (i.e., what are the successful adoption paths for DevOps?). Finally, GT allows researchers to build an independent and original understanding, which is adequate to collect empirical evidence directly from the practice on industry without bias of previous research. The evidence is only reintegrated back with the existing literature after the step of theory construction.

Since the publication of the original version of GT [15], several modifications and variations have been proposed to the method, coming to exist at least seven different versions [24]. Here we chose the classic version, mainly because we did not have a research question at the beginning of our research, exactly as suggested in this version. We actually started from an area of interest: successful DevOps adoption in industry. In addition, research works in software engineering that leverage GT predominantly use this version [22]. We carried out our research using an existing guideline about how to conduct a Grounded Theory [23] research. This guideline organizes a GT investigation in 3 steps: *Open Coding* Data Collection, *Selective Coding* Data Analysis, and *Theoretical Coding.*

(a) **Open Coding Data Collection.** We started our research by collecting and analyzing data from companies that claim to have successfully adopted DevOps. To this end, we have conducted a *raw data analysis* that searches for patterns of incidents to indicate concepts, and then grouped these concepts into categories [22].

(b) **Selective Coding Data Analysis.** In the second step, we evolve the initial set of categories by comparing new incidents with the previous ones. Selective coding starts when a "core category" is identified [22]. The core category is responsible for enabling the integration of the other categories and structuring the results into a dense and consolidated grounded theory [25]. In selective coding, we only considered the specific variables that are directly related to the core category, in order to enable the production of an harmonic theory [16, 26]. Selective coding ends when we achieve a theoretical saturation, which occurs when the last few participants provided more evidence and examples but no new concepts or categories [15].

(c) **Theoretical Coding.** After saturation, we built a theory that explains the categories and the relationships between the categories. Additionally, we reintegrated our theory with the existing literature, which allowed us to compare our proposal with other theories about DevOps. That is, using a Grounded Theory approach, one should only conduct a literature review in later stages of a research, in order to avoid external influences to conceive a theory [27].

Throughout the process, we wrote memos capturing thoughts and analytic processes; the memos support the emerging concepts, categories, and their relationships [27].

Regarding data collection, we conducted semi-structured interviews with 15 practitioners of companies from Brazil, Ireland, Portugal, Spain, and United States that contributed to DevOps adoption processes in their companies. Participants were recruited by using two approaches: (1) through direct contact in a *DevOpsDays* event in Brazil and (2) through general calls for participation posted on DevOps user groups, social networks, and local communities. In order to achieve a heterogeneous perspective and increase the wealth of information in the results, we consulted practitioners from a variety of companies. Table 3 presents the characteristics of the participants that accepted our invitation. To maintain anonymity, in conformance with the human ethics guidelines, hereafter we will refer to the participants as P1–P15 (first column). *We signed a non-disclosure agreement with the investigated companies to use the data only in the context of our study and, therefore, we cannot disclose them.*

Table 3: Participant Profile. SX means software development experience in years, DX means DevOps experience in years, CN means country of work, and CS means company size (S<100; M<1000; L<5000; XL>5000).

P#	**Job Title**	**SX**	**DX**	**CN**	**Domain**	**CS**
P1	DevOps Developer	9	2	IR	IT	S
P2	DevOps Consultant	9	3	BR	IT	M
P3	DevOps Developer	8	1	IR	IT	S
P4	Computer Technician	10	2	BR	Health	S
P5	Systems Engineer	10	3	SP	Telecom	XL
P6	Developer	3	1	PO	IT	S
P7	Support Analyst	15	2	BR	Telecom	L
P8	DevOps Engineer	20	9	BR	Marketing	M
P9	IT Manager	14	8	BR	IT	M
P10	Network Admin.	15	3	BR	IT	S
P11	DevOps Supervisor	6	4	BR	IT	M
P12	Cloud Engineer	9	3	US	IT	L
P13	Technology Manager	18	6	BR	Food	M
P14	IT Manager	7	2	BR	IT	S
P15	Developer	3	2	BR	IT	S

The interviews were conducted between April 2017 and April 2018 by means of Skype calls. The interviews lasted a minimum of 20 minutes, a maximum of 50 minutes, and an average of 31 minutes. Data collection and analysis were iterative so the collected data helped to guide future interviews. Questions evolved according to the progress of the research. We started with five open-ended questions: (1) What motivated the adoption of DevOps? (2) What does DevOps adoption mean in the context of your company? (3) How was DevOps adopted in your company? (4) What were the results of adopting DevOps? (5) What were the main difficulties?

As the analyzes were being carried out, new questions were added to the script. These new questions were related to the concepts and categories identified in previous interviews. Examples of new questions include: (1) What is the relationship between deployment automation and DevOps adoption? (2) Is it possible to adopt DevOps without automation? (3) How has your company fostered a collaborative culture?

With respect to *data analysis*, the interviews were recorded, transcribed, and analyzed. The interviews with participants from Brazil and Portugal were translated from Portuguese into English.

The first moment of the analysis, called open coding in GT, starts immediately after the transcription of the first interview. Open coding lasted until there was no doubt about the core category of the study. Similar to that described by Adolph et al. [27], we started considering a core category candidate and changed later. The first core category candidate was ***automation***, but we realized that this category did not explain most of the behaviors or events in data. The sense of shared responsibilities in solving problems, and the notion of product thinking are examples of events that could not be naturally explained in terms of ***automation***. We then started to understand that ***collaborative culture*** also appeared recurrently in the analysis and with more potential to explain the remaining events. Thus, we asked the respondents explicitly about the role of ***automation*** and how the ***collaborative culture*** is formed in a DevOps adoption process.

Considering the script adaptations and the analysis of new data in a constant comparison process, taking into account the previous analyses and the respective memos written during all the process, after the tenth interview, we concluded that ***collaborative culture*** was the core category regarding how DevOps was successfully adopted. At this moment, the open coded ended and the selective coding started. We started by restricting the coding only to specific variables that were directly related to the core category and their relationships. Following three more interviews and respective analysis, we realized that the new data added less and less content to the emerging theory. That is, the explanation around how the ***collaborative culture*** category is developed showed signs of saturation. We then conducted two more interviews to conclude that we had reached a theoretical saturation, that is, we were convinced there were no more enablers or outcomes related to DevOps adoption, the relationship between all of them was adequate and the properties of core category were well developed.

At this point, we started the theoretical coding to find a way to integrate all the concepts, categories, and memos in the form of a cohesive and homogeneous theory, where we have pointed out the role of the categories as enablers and outcomes. We will present more details about the results of our theoretical coding phase in the next section. It is important to note that *raw interview transcripts* are full of noise. We started the coding by removing this noise and identifying the key points. Key points are summarized points from sections of the interview [28]. For example:

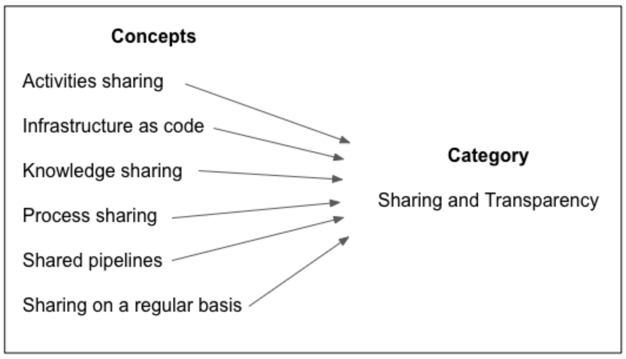

Figure 1: Coding: Building Categories

Raw data: *"So, here we have adopted this type of strategy that is the infrastructure as code, consequently we have the versioning of our entire infrastructure in a common language, in such a way that any person, a developer, an architect, the operations guy, or even the manager, he can look at it and describe that the configuration of application x is y. So, it aggregates too much value for us exactly with more transparency"*

Key point: *"Infrastructure as code contributes to transparency because it enables the infrastructure versioning in a common language to all professionals"*

We then assigned codes to the key point. A code is a phrase that summarizes the key point and one key point can lead to several codes [21].

Code: *Infrastructure as code contributes to transparency*

Code: *Infrastructure as code provides a common language*

In this example, the concept that emerged was "infrastructure as code". The expression corresponding to this concept comes directly from raw data, but this is not a rule. It is common for the concept to be an abstraction, without emerging from an expression present in raw data. At this moment, we already identified other concepts that contribute to transparency. We then wrote the following memo:

Memo: *Similar to sharing on a regular basis and shared pipelines, the concept of infrastructure as code is an important transparency related one. These transparency related concepts have often been cited as means to achieve greater collaboration between teams.*

The constant comparison method was repeated on the concepts to produce a third level of abstraction called categories. Infrastructure as code was grouped together with five other concepts into the **sharing and transparency** category. Figure 1 illustrates how that abstraction of concepts become a category.

2.2. Focus Group

Focus group emerged as a research method in the social sciences in the 1950s and is currently widely used, for example, in sociological studies, market research, product planning, and system usability studies [29]. Morgan [30] defines focus group as a research technique that collects data through group interaction on a specific topic determined by the researcher.

According to F. Shull et al. [29], focus groups typically have between three and twelve participants, are designed to obtain personal perceptions of members of one or more groups involved in a defined area of research interest and have as benefits the production of candid, often insightful information, with a low cost and fast execution. These characteristics make the focus group an adequate alternative to the purposes of assessing our DevOps adoption model. According to the authors, the discussion is guided and facilitated by a researcher-moderator who follows a predefined structure of questions.

We followed a structure similar to that performed by Lehtola et al. [31], consequently, the focus group was conducted as follows: (1) the researcher-moderator served as focus group facilitator providing participants with the discussion topics; (2) at the beginning of each topic' discussion, the questions were presented to participants who wrote their ideas and keywords in post-it notes and (3) after that, the notes were placed on a white board and served as a starting point for discussions on the respective topic in order to reach conclusions about the respective question.

3. DevOps: Categories and Concepts

Here we detail our understanding of the core category of DevOps adoption (***collaborative culture***) and relate it to categories that either work as DevOps **enablers** or are expected **outcomes** of a DevOps adoption process. We have highlighted the concepts along with raw data quotes from the interviews.

3.1. The Core Category: Collaborative Culture

The ***collaborative culture*** is the core category for DevOps adoption. A ***collaborative culture*** essentially aims to remove the silos between development and operations teams and activities. As a result, operations tasks—like deployment, infrastructure provisioning management, and monitoring— should be considered as regular, day-to-day, development activities. This leads to the first concept related to this core category: **operations tasks should be performed by the development teams in a seamless way.**

"*A very important step was to bring the deployment into day-to-day development, no waiting anymore for a specific day of the week or month. We wanted to do deployment all the time. Even if in a first moment it were not in production, a staging environment was enough. [...] Of course, to carry out the deployment continuously, we had to provide all the necessary infrastructure at the same pace.*" (P14, IT Manager, Brazil)

Without DevOps, a common scenario is an accelerated software development without concerns about operations. At the end, when the development team has a minimum viable software product, it is sent to the operations team for publication. Knowing few things about the nature of the software and how it was produced, the operations team has to create and configure an environment and to publish the software. In this scenario, software delivery is typically delayed and conflicts between teams show up. When a ***collaborative culture*** is fomented, teams collaborate to perform the tasks from the first day of software development. With the constant exercise of provisioning, management, configuration and deployment practices, software delivery becomes more natural, reducing delays and, consequently, the conflicts between teams.

"*We work using an agile approach, planning 15-day sprints where we focused on producing software and producing new releases at a high frequency. However, at the time of delivering the software, complications started to appear. (...) Deliveries often delayed for weeks, which was not good neither for us nor for stakeholders.*" (P6, Developer, Portugal)

As a result of constructing a ***collaborative culture***, the development team no longer needs to halt its work waiting for the creation of one application server, or for the execution of some database script, or for the publication of a new version of the software in a staging environment. Everyone needs to know the way this is done and, with the collaboration of the operations team, this can be performed in a regular basis. If any task can be performed by the development team and there is trust between the teams, this task is incorporated into the development process in a natural way, manifesting the second concept related to ***collaborative culture*** category: **software development empowerment**.

" *It was not feasible to have so many developers generating artifacts and stopping their work to wait for another completely separate team to publish it. Or needing a test environment and having to wait for the operations team to provide it only when possible. These activities have to be available to quickly serve the development team. With DevOps we supply the need for freedom and have more power to execute some tasks that are intrinsically linked to their work.*" (P5, Systems Engineer, Spain)

A ***collaborative culture*** requires **product thinking**, in substitution to **operations or development thinking**. The development team has to understand that the software is a product that does not end after "pushing" the code to a project's repository and the operations team has to understand that its processes do not start when an artifact is received for publication. **Product thinking** is the third concept related to our core category.

"*We wanted to hire people who could have a product vision. People who could see the problem and think of the best solution to it, not only thinking of a software solution, but also the moment when that application will be published. We also brought together developers to reinforce that everyone has to think of the product and not only in their code or in their infrastructure*" (P12, Cloud Engineer, United States)

There should be a **straightforward communication** between teams. Ticketing systems are cited as a typical and inappropriate means of communication between development and operations teams. Face-to-face communication is the best option, but considering that it is not always feasible, the continuous use of tools like *Slack* and *Hip Chat* was cited as more appropriate options.

"*We also use this tool (Hip Chat) as a way to facilitate communication between development and operations teams. The pace of work there is very accelerated, and thus it is not feasible to have a bureaucratic communication. (...) This gave us a lot of freedom to the development activities, in case of any doubt, the operations staff is within the* *reach of a message.*" (P5, Systems Engineer, Spain)

There is a *shared* responsibility to identify and fix the issues of a software when transitioning to production. The strategy of avoiding liability should be kept away. The development team must not say that a given issue is a problem in the infrastructure, then it is operations team' responsibility. Likewise, the operations team must not say that a failure was motivated by a problem in the application, then it is development team's responsibility. A **blameless** context must exist. The teams need to focus on solving problems, not on laying the blame on others and running away from the responsibility. The sense of **shared responsibilities** involves not only solving problems, but also any other responsibility inherent in the software product must be shared. **Blameless** and **shared responsibilities** are the remaining concepts of the core category.

"*We realized that some people were afraid of making mistakes. Our culture was not strong enough to make everyone feel comfortable to innovate and experiment without fear of making mistakes. We made a great effort to spread this idea that no-one is to be blamed for any problem that may occur. We take every possible measure to avoid failures, but they will happen, and only without blaming others we will be able to solve a problem quickly.*" (P8, DevOps Engineer, Brazil)

At first glance, considering the creation and strengthening of the ***collaborative culture*** as the most important step towards DevOps adoption seems somewhat obvious, but the respondents cited some mistakes that they consider recurrent in not prioritizing this aspect in a DevOps adoption:

"*In a DevOps adoption, there is a very strong cultural issue that the teams sometimes are not adapted to. Regarding that, one thing that bothers me a lot and that I see very often is people hitching DevOps exclusively by tooling or automation.*" (P9, IT Manager, Brazil)

Besides the core category (***collaborative culture***), we have identified three other sets of categories: the enablers of DevOps adoption, the consequences of adopting DevOps, and the categories that are both enablers and consequences.

3.2. Enabler Categories

Next we detail the categories that support the adoption of DevOps practices, including ***automation*** and ***sharing and transparency***.

3.2.1. Automation

This category presents the higher number of related concepts. This occurs because manual proceedings are considered strong candidates to propitiate the formation of a silo, hindering the construction of a ***collaborative culture***. If a task is manual, a single person or team will be responsible to execute it. Although ***transparency*** and ***sharing*** can be used to ensure collaboration even in manual tasks, with automation the points where silos may arise are minimized.

"*When a developer needed to build a new application, the previous workflow demanded him to create a ticket to the operations teams, which should then manually evaluate and solve the requested issue. This task could take a lot of time and there was no visibility between teams about what was going on (...). Today, those silos do not exist anymore within the company, in particular because it is not necessary to execute all these tasks manually. Everything has been automated.*" (P12, Cloud Engineer, United States)

In addition to contributing to ***transparency***, ***automation*** is also considered important to ensure *reproducibility* of tasks, reducing rework and risk of human failure. Consequently, ***automation*** increases the confidence between teams, which is an important aspect of the ***collaborative culture***.

"*Before we adopted DevOps, there was a lot of manual work. For example, if you needed to create a database schema, it was a manual process; if you needed to create a database server, it was a manual process; if you needed to create additional EC2 (Amazon Elastic Compute Cloud) instances, such a process was also manual. This manual work was time consuming and often caused errors and rework.*" (P1, DevOps Developer, Ireland)

The eight concepts regarding the ***automation*** category will be detailed next. In all interviews we extracted explanations about **deployment automation** (1), as part of DevOps adoption. Software delivery is the clearest manifestation of value delivery in software development. In case of problems in deployment, the expectation of delivering value to business can quickly generate conflicts and manifest the existence of silos. In this sense, ***automation*** typically increases agility and reliability. Some other concepts of automation go exactly around deployment automation.

It is important to note that frequent and successfully deployments are not sufficient to generate value to business. Surely, the quality of the software is more relevant. Therefore, quality checks need to be automated as well, so they can be part of the deployment pipeline, as is the case of **test automation** (2). In addition, to automate application deployment, the environment where the application will run needs to be available. As a consequence, **infrastructure provisioning automation** (3) must be also considered in the process. Besides being available, the environment needs to be properly configured, including the amount of memory and CPU, availability of the correct libraries versions, and database structure. If the configuration of some of these concerns has not been automated, the deployment activity can go wrong. Therefore, the automation of **infrastructure management** (4) is another concept of the ***automation*** category.

"Our primary motivation for adopting DevOps was basically reducing rework. Almost every week we had to basically build new servers and start them manually, which was very time-consuming." (P4, Computing Technician, Brazil)

Modern software is built around services [32]. Microservices was commonly cited as one aspect of DevOps adoption. To Fowler and Lewis [33], in the microservice architectural style, services need to be independently deployable by fully automated deployment machinery. We call this part of microservices characteristics of **autonomous services** (5). **Containerization** (6) is also mentioned as a way to automate the provisioning of containers—the environment where these autonomous services will execute. **Monitoring automation** (7) and **recovery automation** (8) are the remaining concepts. The former refers to the ability to monitor the applications and infrastructure without human intervention. One classic example is the widespread use of tools for sending messages reporting alarms—through SMS, Slack/Hip Chat, or even cellphone calls—in case of incidents. The latter is related to the ability to either replace a component that is not working or rollback a failed deployment without human intervention.

3.2.2. Transparency and Sharing

It represents the grouping of concepts whose essence is to help disseminate information and ideas among all. Training, tech talks, committees lectures, and round tables are examples of these events. Creating channels by using communication tools is another recurrent topic related to ***sharing*** along the processes of DevOps adoption. According to the content of what is shared, we have identified three main concepts: (1) **knowledge sharing**: the professionals interviewed mention a wide range of skills they need to acquire during the adoption of DevOps, citing structured sharing events to smooth the learning curve of both technical and cultural knowledge; (2) **activities sharing**: where the focus is on sharing how simple tasks can or should be performed (e.g., sharing how a bug has been solved). Communication tools, committees, and round tables are the common forum for sharing this type of content; and (3) **process sharing**: here, the focus is on sharing whole working processes (e.g., the working process used to provide a new application server). The content is more comprehensive than in sharing activities. Tech talks and lectures are the common forum for sharing processes.

> "Nowadays ... everyone knows what everyone else is working on. That's why we have some structured actions. For instance, if one person wants to share some content relevant to the rest of the team, she is completely free to book a room and carry out her own tech talk. ... Considering this investment in removing silos, the guy knows that every procedure should be transparent and shared to anyone that is interested on that. This creates a positive cycle that increases the sense of collaboration and makes one to share more details about what she does. "(P7, Support Analyst, Brazil)

Sharing concepts contribute with the ***collaborative culture***. For example, all team members gain best insight about the entire software production process, with a solid understanding of shared responsibilities. A shared vocabulary also emerged from ***sharing*** and this facilitates communication. The use of **infrastructure as code** was recurrently cited as a means for guaranteeing that everyone knows how the execution environment of an application is provided and managed. Next is an interview transcript which sums up this concept.

"*So, here we have adopted this type of strategy that is the infrastructure as code, consequently we have the versioning of our entire infrastructure in a common language, in such a way that any person, a developer, an architect, the operations guy, or even the manager, he can look at it and describe that the configuration of application x is y. So, it aggregates too much value for us exactly with more transparency.*" (P12, Cloud Engineer, United States)

Regarding ***transparency and sharing***, we have also found the concept of **sharing on a regular basis**, which suggests that sharing should be embedded in the process of software development, in order to contribute effectively to transparency (e.g., daily meetings with Dev and Ops staff together was one practice cited to achieve this). As we will detail in the *continuous integration* concept of the ***agility*** category, a common way to integrate all tasks is a pipeline. Here, we have the concept of **shared pipelines**, which indicates that the code of pipelines must be accessible to everyone, in order to foment transparency.

"*The code of how the infrastructure is made is open to developers and the sysadmins need to know some aspects of how the application code is built. The code of our pipelines is accessible to everyone in the company to know how activities are automated*" (P13, Technology Manager, Brazil)

3.3. Categories related to the DevOps adoption outcomes

In this section we detail the categories that correspond to the expected consequences with the adoption of DevOps practices, including ***agility*** and ***resilience***; as discussed as follows.

3.3.1. Agility

Agility is frequently discussed as a major outcome of DevOps adoption. With more collaboration between teams, **continuous integration** with the execution of multidisciplinary pipelines is possible, and it is an agile related concept frequently explored. These pipelines might contain infrastructure provisioning, automated regression testing, code analysis, automated deployment and any other task considered important to continuously execute.

"When we adopted DevOps, our gain in productivity was very large. So we went from a deploy per 15 days to 40 deploys a day. So . . . , we started to delivery more value to the company in small time frames." (P12, Cloud Engineer, United States)

These pipelines encourage two other agile concepts: **continuous infrastructure provisioning** and **continuous deployment**. The latter is one of the most recurrent concepts identified in the interview analysis. Before DevOps, deployment had been seen as a major event with high risk of downtime and failure involved. After DevOps, the sensation of risk in deployment decreases and this activity became more natural and frequent. Some practitioners claim to perform dozens of deployments daily.

3.3.2. Resilience

Also related to an expected outcome of adopting DevOps, ***resilience*** refers to the ability of applications to adapt quickly to adverse situations. The first related concept is **auto scaling**, i.e., allocating more or fewer resources to applications that increase or decrease on demand. Another concept related to the ***resilience*** category is **recovery automation**, that is the capability that applications and infrastructure have to recovery itself in case of failures. There are two typical cases of recovery automation: (1) in cases of instability in the execution environment of an application (a container, for example) an automatic restart of that environment will occur; and (2) in cases of new version deployment; if the new version does not work properly, the previous one must be restored. This auto restore of a previous version decrease the chances of downtimes due to errors in specific versions, which is the concept of **zero down-time**, the last one of the ***resilience*** category.

"When it was necessary to deploy some specific systems, there was often a downtime of a few minutes of the application. In the cases that the deployment did not succeed, the downtime was even greater (perhaps a couple of hours). But with the adoption of DevOps we were able to eliminate the downtime, particularly with the introduction of Kubernetes (https://kubernetes.io/)" (P1, DevOps Developer, Ireland)

3.4. Categories that are both Enablers and Outcomes

Finally, we summarize the categories that are both enablers and outcomes, including ***continuous measurement*** and ***quality assurance***. More details about these categories can be found in the original study [11].

Continuous Measurement. As an enabler, regularly performing the measurement and sharing activities contributes to avoiding existing silos and reinforces the ***collaborative culture***, because it is considered a typical responsibility of the operations team. As an outcome, the continuously collection of metrics from applications and infrastructure was appointed as a necessary behavior of the teams after the adoption of DevOps. It occurs because the resultant agility increases the risk of something going wrong. The teams should be able to react quickly in case of problems, and the continuous measurement allows it to be proactive and resilient.

Quality Assurance. In the same way as continuous measurement, quality assurance is a category that can work both as enabler and as outcome. As enabler because increasing quality leads to more confidence between the teams, which in the end generates a virtuous cycle of collaboration. As outcome, the principle is that it is not feasible to create a scenario of continuous delivery of software with no control regarding the quality of the products and its production processes.

Another two concepts cited as part of quality assurance in DevOps adoption are the use of **source code static analysis** (4) to compute quality metrics in source code and the **parity between environments** (development, staging, and production) to reinforce transparency and collaboration during software development.

4. A Theory on DevOps Adoption

The results of a grounded theory study, as the name of the method itself suggests, are grounded on the collected data, so the hypotheses emerge from data. A grounded theory should describe the key relationships between the categories that compose it, i.e., a set of inter-related hypotheses [21]. We present the categories of our grounded theory about DevOps adoption as a network of the two categories of enablers (***automation***, ***sharing and transparency***) that are commonly used to develop the core category ***collaborative culture***, as discussed in the previous section. Based on our understanding, implementing the enablers to develop the ***collaborative culture*** typically leads to concepts related to two categories of expected outcomes: ***agility*** and ***resilience***. Moreover, there are two categories that can be considered both as enablers and as outcomes: ***continuous measurement*** and ***quality assurance***. In this section, we describe the relationships between those categories, building a theory of DevOps adoption.

4.1. A General Path for DevOps Adoption

In Section 2 we presented the general questions of this research, including: How do practitioners characterize a successfully path for DevOps adoption? Here, we elaborated a response to this question, based on our grunded theory study. The main point that should be formulated is the construction of a ***collaborative culture*** between the software development and operations teams and related activities. According to our findings, the other categories, many of which are also present in other studies that have investigated DevOps, only make sense if the practices and concepts related to them either contribute to the level of a ***collaborative culture*** or lead to the expected consequences of a ***collaborative culture***. This leads to a general hypothesis of our work, which is:

General Hypothesis: *DevOps brings several benefits related to Agility, Resilience, and Software Quality Assurance. Nonetheless, to achieve this benefits, it is highly recommended to work towards a* ***collaborative culture****, removing silos and and implementing a more direct communication between the teams.*

The quote bellow makes explicit the relevance of a ***collaborative culture***—as well as the challenges to keep this culture-to achieve the benefits with DevOps adoption.

"*Keeping (the collaborative) culture alive is still a challenge for us, and we consider it very important. Here in the company, for example, we have tech talks that are monthly conversations with the teams. The purpose of these Tech Talks is to share knowledge, technologies, and work procedures, by increasing the transparency about how everything works. We also have a DevOps culture Slack channel, where we discuss DevOps. The idea is not to let the culture die, . . . , because that is the essence of DevOps for us.*" (P12, Cloud Engineer, United States)

This hypothesis emerged from our understanding about the perceptions of the practitioners on the concepts that might positively influence the adoption of DevOps. We built this understanding iteratively, through discussions considering the transcripts, memos, categories and their relationships. Surelly, during this process, different opinions emerged, though in the end the authors of this paper agreed with this general hypothesis. For intance, in certain moments of the research, we used to believe that DevOps would actually mean the *end of the operations teams.* After several rounds of discussion, we concluded that DevOps actually focus on a collaborative approach for executing development and operation tasks. This understanding induces four sub-hypothesis, as discussed in what follows.

Hypothesis 1: *Certain categories related to DevOps adoption only make sense if used to increase the* ***collaborative culture*** *level. We call this set of categories of* ***enablers****.*

Based on this hypothesis, the maturity of DevOps adoption does not advance in situations where only one team is responsible to understand, adapt, or evolve automation—even when such automation supports different activities like deployment, infrastructure provisioning and monitoring. The same holds for the other *enabling* categories. That is, in situations that ***transparency and sharing*** do not contribute to the ***collaborative culture***, they do not contribute to DevOps adoption as a whole. This is clear when one of the participants of our study states that

"*DevOps involves tooling, but DevOps is not tooling. That is, people often focus on using tools that are called 'DevOps tools', believing that this is what DevOps is. I always insist that DevOps is not tooling, DevOps involves the proper user of tools to improve software development procedures.*" (P2, DevOps Consultant, Brazil)

Hypothesis 2: *Some other categories are not related to DevOps adoption for contributing to increase the* ***collaborative culture*** *level, but insted for emerging as an expected or necessary consequence of the adoption. These represent the set of* ***outcome*** *categories.*

In a first moment, the simple fact that a team is more ***agile*** in delivering software, or more ***resilient*** in failure recovery, does not contribute directly to bringing operations teams closer to development teams. Nevertheless, a signal of a mature DevOps adoption is an increasing capacity for continuously delivering software (and thus being more ***agile***) and for building ***resilient*** infrastructures.

Hypothesis 3: *The categories* **Continuous Measurement** *and* **Quality Assurance** *are both related to DevOps enabling capacity and to expected DevOps outcomes.*

Measurement is cited as a typical responsibility of the operations team. At the same time that sharing this responsibility reduces silos, it is also cited that measurement is a necessary consequence of DevOps adoption. Particularly because the continuous delivery of software requires more control, which is supplied by concepts related to the ***continuous***

measurement category. The same premise is valid to the ***quality assurance*** category. At first glance, ***quality assurance*** appears as one response to the context of agility in operations as a result of DevOps adoption. But, the efforts in quality assurance of software products increase the confidence between the development and operations teams, increasing the level of ***collaborative culture***.

Altogether, DevOps enablers are the means commonly used to increase the level of the ***collaborative culture*** in a DevOps adoption process. We have identified five categories of DevOps enablers: ***Automation***, ***Continuous Measurement***, ***Quality Assurance***, ***Sharing***, and ***Transparency***. Another finding of our study leads to our fourth hypothesis.

Hypothesis 4: *There is no precedence between enablers in a DevOps adoption process.*

We have realized that the adoption process might not have to prioritize any enabler, and a company that aims to implement DevOps should start with the enablers that seem more appropriate (in terms of its specificities). Accordingly, we did not find any evidence that an enabler is more efficient than another for creating a ***collaborative culture***. ***Automation*** is the category that appears more frequently in our study, though several participants make clear that associating DevOps with automation is a misconception.

DevOps outcomes are the categories that does not primarily produce the expected effect of an **enabler**, typically concepts that are expected as consequences of an adoption of DevOps. We have identified four categories that can work as DevOps outcomes: ***agility***, ***continuous measurement***, ***quality assurance***, and ***software resilience***. Note that, as mentioned before, ***continuous measurement*** and ***quality assurance*** are both enablers and outcomes.

That is, a well succeeded DevOps adoption typically increases the potential of ***agility*** of teams and enables ***continuous measurement***, ***quality assurance*** and ***resilience*** of applications. However, in some situations, this potential is not completely used due to business decisions. For example, one respondent cited that, at a first moment, the company did not allow the continuous deployment (more potential of agility) of applications in production.

Considering the hypothesis we build from our understanding about a successfully path for DevOps adoption, we answer our first research question "*(RQ1) How do practitioners characterize a successfully path for DevOps adoption?*"

(RQ1) Answer: In order to successfully conduct an effort for DevOps adoption, practitioners suggest the focus on building a ***collaborative culture*** (this is the essence of DevOps), which should be achieved through Automation, Transparency and Knowledge Sharing, Continuous Measurement, and Quality Assurance. Without setting up ***collaborative culture*** as the main goal, the chances of failing to achieve the expected benefits of adopting DevOps (e.g., Agility and Resilience) increase.

4.2. Outline of the Theory on DevOps Adoption

We summarize our theory in this section using a set of recommendations about building and reporting theories in software engineering [13]. Accordingly, a theory should be described in terms of **constructs**, **propositions**, **explanation** for the propositions, and **scope** of the theory.

The main construct of our theory is *DevOps adoption*, which means any effort for building a ***collaborative culture*** between the development and operations teams. DevOps adoption is supported by other constructs, including *automation* (deployment automation, infrastructure provision automation, test automation, and so on) and knowledge sharing (e.g., sharing procedures and making clear the task assignments of the teams). We augmented our hypothesis to identify a set of nine propositions, which we can explain by means of the categories, categories' relations, and transcription memos. For instance, the necessity of using tech talks, simplifying communication procedures, and employing tools like Slack and Hip Chat explains the relevance of knowledge sharing (P2). Similarly, we can explain proposition (P7) by considering our research transcriptions, like "*DevOps reduces (or even eliminates) downtime*". The scope of our theory relates to enterprise systems (in particular those based on a service-oriented architecture).

(P1) The use of automation enables DevOps

(P2) The use of practices and tools for sharing knowledge enables DevOps

(P3) The use of continuous measurement procedures enables DevOps

(P4) The use of quality assurance methods enables DevOps

(P5) DevOps centered on ***collaborative culture*** increases the agility of the teams

(P6) DevOps centered on ***collaborative culture*** increases systems' resilience

(P7) DevOps centered on ***collaborative culture*** decrease systems' downtime

(P8) DevOps centered on ***collaborative culture*** supports continuous measurement

(P9) DevOps centered on ***collaborative culture*** supports SQA activities

Figure 2 represents the elements of our theory using the notation introduced in [13]. In this notation, a construct is either represented as a class (light grey boxes) or as an attribute of a class (white boxes within classes). Relationships model the propositions as arrows, and the direction of the arrow models a cause-effect relation. For instance, in Figure 2, the *DevOps Collaborative Culture* increases the agility of both development and operations teams (proposition (P5).

5. A Model for DevOps Adoption

Based on H1-H4 hypothesis, we present a three step model that explains how to adopt DevOps according to our understanding. The model considers the following steps:

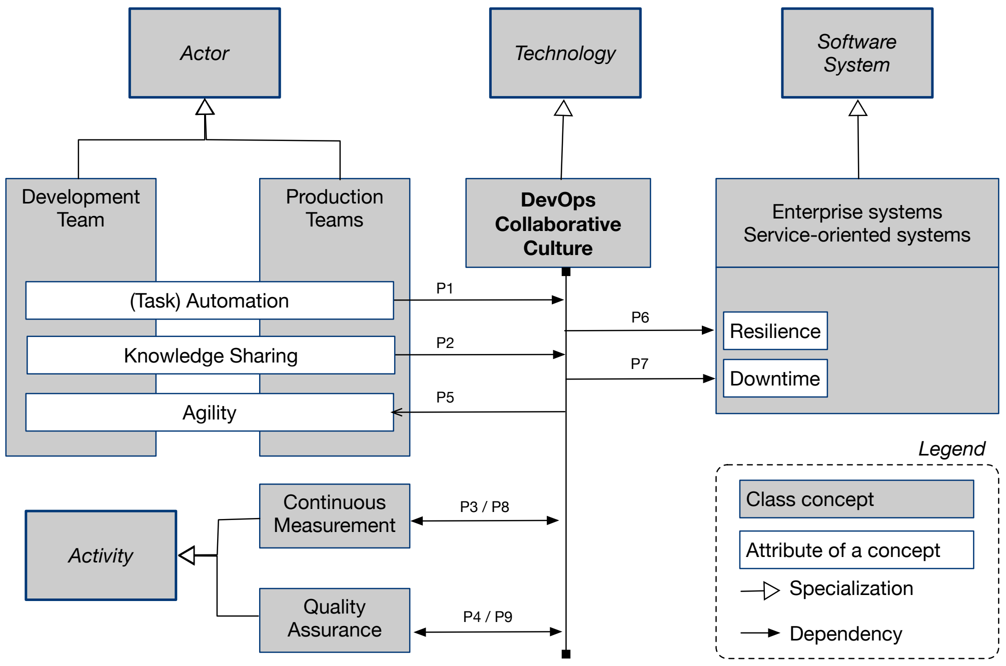

Figure 2: A Theory for DevOps Adoption

1. In the first step, a company should **disseminate the relevance of building a collaborative** culture between development and operations teams, as the main goal of a DevOps effort.

2. In the second step, a company should **select and develop the most suitable enablers** according to its context. The enablers are means commonly used to develop the collaborative culture and its concepts.

3. In the third step, a company should **check the outcomes of the DevOps adoption** in order to verify the alignment with industrial practices and to explore them according to the company's need.

Figure 3 illustrates the categories and the relationships according to the hypothesis. The proposed model is built upon the hypothesis and is one of possible applications of the theory. It was proposed assuming that following the patterns identified in companies that were well succeeded in DevOps adoption can be a good way to reduce the risks of failure during the process.

Our proposed model has been applied to guide the DevOps adoption at the Brazilian Federal Court of Accounts (TCU) where one of the authors of this study works as a software developer. In the next we present the details about the DevOps adoption at TCU and explain the applicability and relevance of our model in a practical scenario.

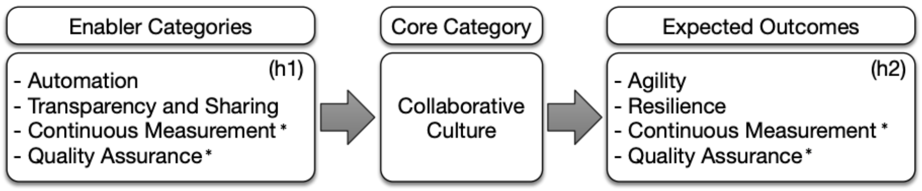

Figure 3: Categories and Relationships. Categories label with a * means thant they are within both *enabler* and *expected outcomes* categories.

6. Application of the Model

In this section we detail a real experience on DevOps adoption where we could demonstrate the practical application of our DevOps adoption model. Initially, we explain the context of the institution where we applied our model; after that, we present our perceptions about the model application; and, finally, we present the results of a focus group containing the perception of four professionals from the institution about the adoption of DevOps as a whole, including their opinion about the applicability and utility of our model during the process.

6.1. Context

TCU is responsible for the accounting, financial, budget, performance, and property oversight of federal institutions and entities of the country. Currently, there are 2500 professionals working at TCU, of which approximately 300 work directly on either software development or operations. The source code repository at TCU hosts more than 200 software projects, totaling over 4 million lines of code.

For many years, the software development process at TCU was characterized by a strong call for standardization of procedures. This approach favored the specialization of activities, causing segregation of development and operations teams in a rigid silo structure. Work on this structure revolved around bureaucratic communication and well-defined service level agreements (SLAs). Communication between teams occurred basically through the use of a corporate service-desk tool. The SLAs mostly ensured deadlines for the operations team to perform the procedures that the development teams requested through the service-desk. The procedures include provisioning of infrastructure and database servers for development, testing, staging, and production environments, requesting access to server logs, and the most controversial of all activities: deployment of new versions of applications. The deployment involved a long chain of a week-long process, whose complex chaining of responsibilities often ran out of schedule, causing months of delays in software deliveries.

In addition to delay the software delivering, the silo structure caused a devastating blaming game between the teams. If there were delays in the delivery process, or errors in some functionality, the focus was not on solving the problem, but rather on pointing out which team was responsible for the problem.

When searching of solutions to mitigate these problems, and knowing the alleged benefits of the approach, TCU included the development of the DevOps approach among the objective of its IT area. The first attempt by the technical teams to reach the goal was to hire consulting firms whose proposed solutions invariably involved implanting specific DevOps tools. These DevOps tools were typically related to automate some aspect of the process and, naturally, the teams turned their attentions to establish clear and separated responsibilities to each aspect of the tools. Obviously, these DevOps tools only changed the points of conflict. In this context, our research was developed and, after that, we applied our model aiming to increase the DevOps level at TCU. In this section we described this experience.

6.2. Applying Our Model for DevOps Adoption

Before applying our model, TCU had been using automation for supporting deployment activities and focusing on the tooling dimension of DevOps. Considering this incomplete perspective, the conflicts between development and operations teams continued. That is, the mere advance in implanting "DevOps tools" simply changed the points of conflict, but they persisted. Here we present our perception about how our model has contributed to DevOps adoption at TCU.

6.2.1. Disseminating the Collaborative Culture

To start disseminating our model at TCU, we conducted a series of lectures to explain both the model itself and the way it was formulated. After knowing the model, the professionals began to understand that if TCU wants to succeed in DevOps adoption, tooling and automation would not be enough. We consider that the teams changed their focus to build a ***collaborative culture***. This change was only possible due to two aspects: (1) the engagement and sponsorship of the IT managers and (2) the explanations about the process of constructing the model: although TCU is a governmental institution, its IT professionals recognize the importance of staying in line with industry practice to avoid an already before faced weight of dealing with completely legacy software.

Looking to the concepts within the ***collaborative culture*** category, the first practical action at TCU was to facilitate communication between teams. The use of tickets and SLAs were then abolished in most of the scenarios. In its turn, the concepts of **software development empowerment** and **shared responsibilities** have found some resistance from some professionals. The strategy to mitigate this cultural resistance involved the use of enablers (that will be further discussed) and awareness about: (1) fortifying the collaboration will bring benefits to all involved and especially to TCU and (2) following a model built upon well succeeded experiences is a low-risk strategy for TCU to explore the benefits of DevOps.

Finally, we consider that the efforts to disseminate and promote the ***collaborative culture*** are continuous, but the kick-off was given and it is possible to notice that the idea has been widely accepted and applied by most of the involved professionals.

6.2.2. Applying the Enablers

Considering the enablers dimension of our model, TCU is currently applying ***sharing and transparency*** concepts. The role of internal tech talks and committees to disseminate that ***collaborative culture*** and related concepts is being reinforced. When a new infrastructure had to be provided and configured, the current guideline is that these activities should be conducted in "pairs" involving members of development and infrastructure teams. All application related tasks must be executed in a collaborative way. Naturally, the professionals noticed that automation would facilitate the operation of that collaboration. For this reason, the infrastructure provisioning and management was automated.

TCU also uses continuous measurement and quality assurance concepts as enablers of its DevOps adoption. The applications started to be continuously tested and measured. The tests were automated and included in the pipelines. Verification of test coverage and quality code also became part of the pipeline. This increased the confidence between teams. More confidence leads to a more robust ***collaborative culture***, and a more robust ***collaborative culture*** enables the company to explore the benefits of DevOps as will be seen below when checking the outputs.

It is important to note that, before DevOps, deployment activities were historically a controversial point at the TCU. Several conflicts occurred over time. Rigid procedures were created to try to avoid problems. However, these "rigid procedures" often led to periods of months without any software delivery. As our model advocates that there is no precedence between enablers, the specific characteristics of the company are a good starting point to select the most relevant enablers to explore. The TCU teams started by the deployment, applying automation and quality checking in the process. With the automated execution of tests and static source code analyzer, the development team demonstrated to the operations team that the products had quality, increasing confidence between them and reducing the necessity of rigid procedures to put some software in production. To automate the deployment in a collaborative way, the teams jointly developed an infrastructure as code strategy to put the details about deployment in the application source code repository itself, increasing the transparency. Little by little the deployment of applications has ceased to be a ceremonial event to become a day-to-day activity.

6.2.3. Checking the Outcomes

Initially, we highlight that a more robust ***collaborative culture*** enabled TCU to explore a scenario of continuous deployment of some applications. The newer applications have pipelines that publish every commit in production. This continuous delivery is only possible in applications whose pipelines contain several quality checks, mitigating the risk of something going wrong. Additionally, automated monitoring is required for these applications. Some applications have a history of dozen of deployments in a single day, contrasting with one deployment per week in the best case, or more than a month without publishing in the worst.

Currently, among the TCU's 131 enterprise systems, 15 are developed in a DevOps way. To illustrate the benefits of DevOps in terms of continuous delivery, at Table 4 we present some numbers related to deployments in production of the most significant systems. Considering the e-TCE system, after the DevOps adoption, it was possible to make 29 deployments on a single day. Before the DevOps adoption, and due to the rigid policies of the operations team, the deployments were schedule to occur once a week. Now, for these specific systems, it is possible to deploy at any moment and the typical downtime of the old enterprise system (around 15 minutes) was almost completely eliminated for the new systems using Kubernetes. That is, making the deployment activities more agile and reducing the downtime of the applications are the current main outcomes of introducing DevOps at TCU.

System Name	**MDDS**	**MWDS**
Autenticidade de Documentos (26 KLOC)	18	37
Cobrança Executiva (33 KLOC)	12	33
Conecta TCU (39 KLOC)	42	51
e-Cautelares (47 KLOC)	9	12
e-TCE (261 KLOC)	29	68
e-TCU Gestores (98 KLOC)	18	64
Mapa de Exposição a Fraude (7 KLOC)	9	39
Ministro (48 KLOC)	17	51
Siga (55 KLOC)	12	17

Table 4: Deployments of TCU' Enterprise Systems. MDDS means the maximum number of deployments in a day. MWDS means the maximum number of deployments in a week

To reduce even more the risks related to continuously delivery software in production, and taking advantage of greater collaboration between its development and operations teams, the TCU started to explore the potential of DevOps tools, like recovery automation, zero down-time, and auto scaling.

In our evaluation, the TCU' change of focus from tooling to collaboration was decisive to increase the confidence between teams, reduce the blame game and, consequently, to enable the company to sustainably exploit the benefits of adopting DevOps. In addition, our model has served as a roadmap that allows the teams to focus on the ***collaborative culture***. So, the outcomes already present in TCU' DevOps Adoption are compatible with those foreseen in the third step of our model.

6.2.4. General Considerations

Since the TCU is a government institution, some advances in DevOps adoption still comes up against regulatory issues. For example, there are internal regulations that establish that only the operations sector is responsible for issues related to application infrastructure, contrasting with shared responsibilities that are part of the ***collaborative culture***.

The ***collaborative culture*** needs to be continuously fomented and our model is only a general roadmap, the efforts need to continue over time to deal with incidents that disfavor the collaboration and to restrain the emergence of new ones.

Nevertheless, our model enabled the TCU to adopt DevOps in a more sustainable way. Knowing the role of each DevOps element in the adoption was fundamental for the TCU to avoid points of failure and to build a collaborative environment that supports the exploration of DevOps benefits.

6.3. Assessment of our DevOps Adoption Model

To reduce the bias of our perception about the scenario of adopting DevOps at TCU, we conducted a Focus Group with four professionals of the company for an empirical evaluation of their perception about DevOps adoption.

The focus group at TCU lasted approximately 3 hours and was attended by four professionals, two of each development and operations team. The participants' profile is described in Table 5.

P#	**Team**	**Educational background**	**Experience**
P1	Dev	Graduate	3 years in dev team at TCU and 9 years of previous experience
P2	Dev	Posgraduate	6 years in dev team at TCU and 7 years of previous experience
P3	Ops	Graduate	3 years in ops team at TCU and 8 years of previous experience
P4	Ops	Graduate	3 years in ops team at TCU and 10 years of previous experience

Table 5: Focus Group Participants

We approached three discussion topics during the focus group. These topics are listed in Table 6 and its results are presented in the remaining of this section.

6.3.1. Current Status of DevOps Adoption at TCU

The first action indicated and discussed in the group was the **provision of environments** (Virtual Machines, VM for short) for the installation of tools that are related to the development work. This action was exemplified by the successful installations of the Elasticsearch[2] and Kafka[3] tools. The previously existing problem was that when a developer had a need of tools like these, he/she would have to open a request for the operations team to provide it—with very long deadlines that often make the use of the most adequate solution unfeasible. With VM provisioning and cooperation between the two teams, these tools became available quickly for use and the responsibility for its management is joint. This is a clear example of application of the concepts of **software development empowerment** and **shared responsibility** of the core category ***collaborative culture***.

Next, the use of **microservices** and **containers** as environment to run them was debated. The first problem that this action solved, in the participants' understanding, was the previous lack of parity between the environments (development, test, staging and production). The recurring problems of applications that worked in a development environment

[2]https://www.elastic.co/
[3]https://kafka.apache.org/

	Topic	Questions
1	Current status of DevOps adoption at TCU	1. What actions developed in the TCU do you consider to be part of DevOps adoption?
		2. What previously existing problems have been solved by these actions?
2	Applicability and utility of the proposed model	1. Do you consider that the proposed model has contributed to DevOps adoption at TCU?
		2. If so, what are the main contributions?
3	Challenges faced and next steps in DevOps adoption	1. What are the main challenges that TCU currently faces in DevOps adoption?
		2. What are the next steps in DevOps adoption at TCU?

Table 6: Focus Group Topics

but which had problems in production were recalled, which was solved with the use of containers. In this context, the use of tools like Docker[4] and Kubernetes[5] was discussed as means of providing configuration details of the environments in the applications source code repositories themselves. This enabled both development and operations teams to get a first idea of the running environment of each application in a more transparent way. Finally, the use of containers and related tools also enabled the use of mechanisms for horizontal scalability, high availability and publication of applications without down-time, solving a recurring problem of interrupting the work of business teams during the deployment of the applications. Here, it is possible to identify several concepts of our theory, such as: (1) parity between environments, (2) infrastructure provisioning automation, (3) autonomous services, (4) containerization, (5) auto scaling, (6) recovery automation and (7) zero down-time.

The third discussed point was the **reduction of bureaucracy in communication** between the teams. It was pointed out that although there is still much room for progress, this can already be considered as one of the advances of the TCU related to DevOps adoption. During the discussion, the ceremonious communication process in the scope of the deployment SLA was recalled. There is a current guideline to avoid using service-desk for simple problem solving. The problems had to be solved in a collaborative way, preferably face to face and the use of the Slack tool has been institutionalized and facilitated the contact between the two teams. We highlight the concept **straightforward communication** of the ***collaborative culture*** category as part of this point of discussion.

Then, the use of **multidisciplinary pipelines** in the most recent applications of TCU

[4]https://www.docker.com/
[5]https://kubernetes.io

was pointed out and debated. These pipelines involve everything from the build, through automated tests and static analysis of source code, execution of containers using Kubernetes and publication isonomically in the different environments (development, staging and production). Jenkins is used as a tool to describe and execute the pipelines. One single trigger execute a set of steps that previously required several comings and goings between the teams and a long time to complete. The group agreed that the construction of multidisciplinary and collaboratively produced and maintained pipelines like these ones is only possible when the ***collaborative culture*** is fostered. This point of discussion refers to a few more concepts: (1) operations in day-to-day development, (2) test automation, (3) deployment automation, (4) shared pipelines, (5) continuous integration, (6) continuous infrastructure provisioning, (7) continuous deployment, (8) continuous testing and (9) source code static analysis.

Automated Database Migrations was the next point considered as part of DevOps adoption. The participants explored the differences between the previous scenario — where changes in the database structure of an application needed to be made according to an SLA and, therefore, requested through service-desk — and the new one where the Flyway[6] tool is used to manage database migrations. The use of these type of tool had previously been discarded because the operations team could not provide the database owner password. This discussion was retaken recently, and the teams jointly developed a solution to safely share the owner password. This discussion reinforces two aspects of the ***collaborative culture*** that are the confidence and the collaboration between the teams. The concept of **infrastructure management automation** was present here.

Finally, the last point considered by the group as part of DevOps adoption was one more solution built in a collaborative way for **continuously and automated monitoring** of application errors. Previously, a simple access to an application's log needed to be requested through the service-desk. The solution automatically collects the logs, searches for errors in its content, and, in case of errors, sends messages through Slack to both teams. We can find the application of two other concepts: (1) monitoring automation and (2) application log monitoring.

6.3.2. Applicability and Utility of the Proposed Model

All participants of the focus group agreed that the proposed model has great utility in DevOps adoption at TCU. They remembered that most of the actions discussed in the previous topic were direct result of the model development and, therefore, its application is already being effective and producing results in expanding DevOps usage throughout the development of TCU' enterprise applications. The following are the two main benefits of model usage, discussed in the focus group:

DevOps Institutional Understanding: In response to the question about the model contributions to TCU, initially it was pointed out that, during the initial attempts with consultant firms, it was clear that the mere use of tools did not bring the teams closer together. Some developers acted as if DevOps had given them permission to ignore operations

[6]https://flywaydb.org/

team' procedures. The operations team, on the other hand, was overly concerned in formally delimit the administration responsibilities to each used tool. The discussion showed that they all agreed that fostering a ***collaborative culture*** was not taken into account before. Throughout the model, the DevOps adoption in well succeeded scenarios goes mainly through this point, has made possible a change in teams posture about collaboration.

Subsequently, a post-it note about the "wide range of practices and experiences" present in the model was discussed. It was once again recalled that several practices have already been implemented using as input industry experiences collected during model production. The model has also been pointed out as a tool for evaluating practices that the TCU does not yet adopt, providing a road map to guide next steps.

Industry Experiences: Finally, it was emphasized that the model was built taking into account successful industry experiences and this represents great value for the TCU. In the group's understanding, although the TCU possesses many governmental environments peculiarities, the search for technological innovation is part of its strategic map, and cannot be achieved by looking only at scenarios similar to the current one. It was emphasized that the industry is an important player in the definition of new technologies that, adapted to a greater or lesser degree, may be fully applicable to government agencies, such as TCU. According to the formed understanding, the fact that this model, built upon industry experiences, is already being effectively applied, is one more confirmation that government agencies can effectively innovate following industry tendencies.

6.3.3. Challenges Faced and Next Steps in DevOps Adoption

The discussions of the last focus group topic focused on identifying the challenges faced during the evolution of DevOps usage at TCU, as well as the next steps to overcome the challenges and institutionalize DevOps as a software development approach.

DevOps Internal Understanding Maturity: It was initially debated the perception pointed out by one of the participants that there is still a lot of hype around what would be DevOps adoption. According to him, some developers still thinking that DevOps allows them to take technical initiatives without consulting other professionals, and that some operations people still do not feel comfortable with this paradigm shift because they understand that DevOps can cause disorganization in an environment that already had stability. It was mentioned that the model helps to deal with this challenge, but that a permanent effort is needed to foment the collaboration between the teams, avoiding someone to leave wanting to solve everything according to personal convictions.

It was also pointed out as a challenge, the difficulty of disseminating knowledge related to new tools and processes that came along with DevOps adoption. Actions to mitigate this challenge have been discussed, including the expansion of internal lectures, participation in events such as DevOpsDays, and the stimuli that the TCU already offers to its professionals, such as training license, refund of training, and availability of one online training platform. In this sense, it was understood that one of the next steps is the expansion of the technical capacity of the professionals in themes related to the modernization of tools and processes.

Information Security: Here, the group discussed that DevOps adoption has considerably increased the TCU's surface of technological vulnerabilities. It was pointed out that the operations team is very concerned about information security, and that its professionals are evaluating the implemented tools and will propose modifications. The participants then aligned that this debate cannot be only on the operations team, as this is a manifestation of collaboration lack.

The P2 then suggested that these concerns extend the scope from DevOps to a DevSecOps context, when security activities are also integrated into the development process. There is then a further step, which is to extend the DevSecOps perspective.

Metrics Collection in Applications: In this part of the discussion, the participants discussed that the current continuous monitoring solution is restricted to application errors, and that the model contains ideas about collecting metrics in applications to foster business decisions and applications evolutions. The P4 pointed out that the same solution can be extended as long as the applications were instrumented to generate logs of any other metrics. Therefore, the continued collection of other application metrics has been pointed out as one of the next steps in adopting DevOps at TCU.

Regulatory Issues: Here, the group understood that, although the model has made it possible to understand that the most important thing is to foster the ***collaborative culture***, many professionals still thinking in a legalistic perspective, and the internal regulations about the TCU' organizational structure establish that the responsibilities for issues related to the application infrastructure are from its operations team, which makes it difficult to consolidate a sense of shared responsibility.

There was no consensus about the best solution to resolve the constraints contained in organizational structuring regulations. Some (P1 and P4) understand that it would be appropriate part of the operations team to be transferred to development sector. Others (P2 and P3) have demonstrated the understanding that a change in the regulations is enough to define that there is shared responsibility for issues related to application infrastructure. There is a working group constituted in order to propose modifications in the regulations to adjust these assignments to the DevOps scenario.

Physical Distancing of Teams: The last challenge discussed during the focus group was the existence of separate rooms for the development and operations teams. Physical distance has been put as a factor that hinders the communication and the developing of the ***collaborative culture***. The participants agreed that the physical approximation of the teams involves questions related to restructuring regulations, as discussed above. If the operations team is incorporated into the development team, the approximation is likely to occur, otherwise it is necessary to seek another viable solution.

6.3.4. General Considerations

Even though it was not the only point of debate, considerations about the model permeated all the discussed topics, practical actions were highlighted that only materialized due to the exchange of experiences that occurred during this research. Many of the concepts presented in the model were visualized during focus group discussions, this is not a mere coincidence and emphasizes that our model is guiding, in a high level of abstraction, the actions of the TCU toward DevOps adoption. In addition, it was possible to note a previously non-existent concern about the ***collaborative culture***, the participants frequently placed their actions as part of the efforts to foster the ***collaborative culture***. Altogether, the results of the focus group allow us to answer our second research question *(RQ2) How does our model contribute to the adoption of DevOps on a specific scenario?*

(RQ2) Answer: The DevOps adoption model changed the focus from automation to a ***collaborative culture***, during the TCU experience on DevOps adoption. This changing has reduced the distance between both teams (development and operations) and eliminated mechanisms (such as ticket systems) that actually decrease the performance of the teams. In addition, the set of perceptions from practitioners, in which our model builds upon, is guiding the decisions on the adoption process at TCU.

7. Threats to Validity

Regarding construct validity, we are actually relying on the subjective practitioners' perception when we stated that we performed our study considering successful cases of DevOps adoptions. However, currently, there is no objective way to measure whether or not a DevOps adoption was successful. Although Grounded Theory offers rigorous procedures for data analysis, our qualitative research may contain some degree of research bias. Certainly, other researchers might form a different interpretation and theory after analyzing the same data, but we believe that the main perceptions would be preserved. This is a typical threat related to GT studies, which do not claim to generate definitive findings. The resulting theory, for instance, might be different in other contexts [34].

For this reason, we do not claim that our theory is absolute or final. We welcome extensions to the theory based on unseen aspects or finer details of the present categories or potential discovery of new dimensions from future studies. Future work can also focus on investigating contexts where DevOps adoption did not succeed, aiming to validate if our model could be relevant in this scenario too. Finally, regarding external validity, although we considered in our study the point of view of practitioners with different backgrounds, working in companies from different domains, and distributed across five countries, we do not claim that our results are valid for other scenarios—although we almost achieved saturation after the 12^{th} interview. Accordingly, our degree of heterogeneity complement previous studies that mostly focus in a single company (as we will discuss next).

The focus group was moderated by one of the researchers, the participants were arbitrarily invited, without a general call, and they are co-workers of one of the researchers. Although they were chosen arbitrarily, the choice was made precisely by the prior knowledge of which professionals were directly involved in DevOps adoption at TCU. To mitigate this threat, the participants were informed the purpose of the group was to obtain an evaluation of the DevOps adoption as a whole and that they had total freedom to expose their real opinions, whether they were favorable or not to the implanted model.

8. Related Work

The research literature is particularly rich when it comes to DevOps-related works (e.g., [1, 3, 10]). In a literature review, Erich et al. [9] presents 8 main concepts related to DevOps: culture, automation, measurement, sharing, services, quality assurance, structures and standards. The authors pointed out that the first four concepts are related to the CAMS framework, proposed by Willis [35]. The paper concludes that there is a great opportunity for empirical researchers to study organizations experimenting with DevOps. Other studies (e.g., [1, 2, 3, 4, 10]) mixed literature reviews with empirical data to investigate DevOps. Although our research and recent literate are interested in understanding DevOps, there are subtle differences in both (1) the methodological aspects and (2) the focus of each work.

First of all, none of the aforementioned works focused on explaining the process of DevOps adoption, in particular, using data collected in the industry. This is unfortunate, since the practitioners' perception present an unique point of view that researchers alone could hardly grasp. Moreover, although the literature has a number of useful elements, there is a need to complement such elements with a perspective on how DevOps has been adopted, containing guidance about how to connect all these isolated parts and then enabling new candidates to adopt DevOps in a more consistent way. For instance, the work of Erich et al. [10] focus on investigating the ways in which organizations implement DevOps. However, this work relies only in literature review and does not formulate new hypothesis about DevOps adoption. Second, in terms of results, our main distinct contribution is to improve the guidance to new practitioners in DevOps adoption. Next, we present the overlappings of our results with the existing literature, presenting also the main differences that make the contributions of our work clearer.

The work of Smeds et al. [1] uses a literature review to produce one explanation about DevOps through a set of enablers and capabilities. Additionally, their results present a set of impediments of DevOps adoption based on an interview with 13 subjects of a same company, and whose DevOps adoption process was at an initial stage. The main similarities with our study are: (1) grouping elements as DevOps enablers; and (2) the presence of several similar concepts: (a) testing, deploying, monitoring, recovering and infrastructure automation; (b) continuous integration, testing and deployment; (c) service failure recovery without delay; and (d) constant, effortless communication. The main differences are: (1) their work does not group concepts into categories, for example: most of their enablers were grouped together by us within the ***automation*** category; (2) presents cultural enablers as common contributor to DevOps, not as the most important concern; and (3) the empirical part of the study focus on building a list of possible impediments to DevOps adoption, not on providing guidance to new adopters.

In the study of Lwakatare et al. [2], the authors aimed at characterizing the and formalize what DevOps is about. Through a sequence of interviews, the authors observed the need of four dimensions to compose DevOps, including collaboration, automation, measurement, and monitoring. In a follow up study, Lwakatare et al. [3] proposed a conceptual framework to explain "DevOps as a phenomenon". The framework is organized around five dimensions (collaboration, automation, culture, monitoring and measurement) and these dimensions are presented with related practices. These two works have good similarities with our study. For instance, all aforementioned dimensions are also presented here. The main differences are: (1) collaboration and culture are presented by us as a single abstraction; (2) Concepts related to monitoring and measurement are grouped by us in a single category: ***continuous measurement***; and (3) it does not indicate a major dimension (aka, the core category). Moreover, our work greatly expand the notion of DevOps, proposing a theory for adoption, and indeed applying this theory in a real setting.

França et al. [4] present a DevOps explanation produced by means of a multivocal literature review. The data was collected from multiple sources, including gray literature, and analyzed by using procedures from GT. The results contain a set of DevOps principles, where there is most of the overlapping with our study. In addition, the paper presents a definition to DevOps, issues motivating its adoption, required skills, potential benefits and challenges of adopting DevOps. The main similarities are: (1) Automation, sharing, measurement and quality assurance are presented as DevOps categories; and (2) Their social aspects category is similar to our ***collaborative culture*** category. The main differences are: (1) it presents DevOps as a set of principles, different from enablers and outcomes in our study; and (2) the Leanness category is not present in our study and the ***resilience*** category is not present in theirs; and (3) it does not indicate a core category.

The study conducted by Erich et al. [10], similarly to the others cited above, combined literature review with some interviews with practitioners. In the literature review part, the papers were labeled and the similar labels are grouped. The 7 top labels are then presented as elements of DevOps usage in literature: culture of collaboration, automation, measurement, sharing, services, quality assurance and governance. After the literature review, six interviews were conducted in order to obtain evidence of DevOps adoption in practice. The interviews were analyzed individually and a comparison between them was made, focusing on problems that organizations try to solve by implementing DevOps, problems encountered when implementing DevOps and practices that are considered part of DevOps. The main similarity with our study is that 5 of their 7 groups are also present in our study (culture of collaboration, automation, measurement, sharing and quality assurance). The main differences are: (1) it does not consolidate the practitioners' perspective, but only compare it with literature review results; and (3) it does not indicate a major group.

Finally, the work of Vergori and colleagues [36], the authors proposed a set of metrics related to DevOps performance, such as expected task completion rate, expected finishing time, and the proportion of time doing dev or ops activities. Although some of the metrics proposed are reasonable straightforward to measure (e.g., the task completion rate), some other are not so easy. In this work, the authors used the Phoenix project [37], which is a case of industrial DevOps adoption. In terms of similarities to our work, both works focus on improving DevOps experience. While we focus on DevOps adoption, their work focus on DevOps performance.

In comparison with our previous paper [11], here we advance further in exploring real scenarios of DevOps usage. The TCU scenario was described in details through the results of the focus group. Besides that, we present a more detailed explanation about the application of our model, highlighting each step and presenting some numbers.

9. Final Remarks

In this paper, grounded in data collected from successfully DevOps adoption experiences, we present a theory on DevOps adoption, a model of how to adopt DevOps according to this theory, and a case of applying it in practice.

We found out that the DevOps adoption involves a very specific relationship between seven categories: ***agility***, ***automation***, ***collaborative culture***, ***continuous measurement***, ***quality assurance***, ***resilience***, ***sharing and transparency***. The core category of DevOps adoption is the ***collaborative culture***. Some of the identified categories (i.e., automation and sharing and transparency) propitiate the foundation of a ***collaborative culture***. Other categories (i.e., agility and resilience) are expected consequences of this formation. Finally, two other categories (i.e., continuous measurement and quality assurance) work as both foundations and consequences. We call the foundations categories "DevOps enablers", and the consequences categories "DevOps outcomes". Crucially, this model simplifies the understanding of the complex set of elements that are part of DevOps adoption, enabling it to be more direct and to offer a lower risk of focusing on wrong things. We experimented with this model in real settings, improving the benefits of adopting DevOps within a government institution that faced many problems with the separation between the development and operations teams.

Now that we have our theory, our model, and an initial instantiation of this model in practice, we expect further research explorations. First, we believe that our model could help practitioners that are willing to migrate to a DevOps landscape, but are unsure about how to start. Moreover, our model can provide initial light on the chances that are needed to a company achieve DevOps. For instance, if a company has already a good notion of the enablers of its context, it may suggest that the DevOps adoption could be done more smoothly. Our model and our experience can be seen as an initial guideline in this direction. As a consequence, we expect further instantions in other software companies. These other instantiations are valuable to understand the limits and challenges of our current model, that would only become clear when exercized in different environments, under different time, cost, or personel constraints. In our own research agenda, we plan to propose a list of key performance indicators to assess the success of the DevOps transition. In particular, one question that might be worth investigating is how to measure the impact of DevOps.

References

[1] J. Smeds, K. Nybom, I. Porres, DevOps: A definition and perceived adoption impediments, in: C. Lassenius, T. Dingsøyr, M. Paasivaara (Eds.), Agile Processes in Software Engineering and Extreme Programming, Springer International Publishing, Cham, 2015, pp. 166–177.

[2] L. E. Lwakatare, P. Kuvaja, M. Oivo, Dimensions of DevOps, in: C. Lassenius, T. Dingsøyr, M. Paasivaara (Eds.), Agile Processes in Software Engineering and Extreme Programming, Springer International Publishing, Cham, 2015, pp. 212–217.

[3] L. E. Lwakatare, P. Kuvaja, M. Oivo, An exploratory study of DevOps extending the dimensions of DevOps with practices, ICSEA'16, 2016, pp. 91–99.

[4] B. B. N. de França, H. Jeronimo, Junior, G. H. Travassos, Characterizing DevOps by hearing multiple voices, in: Proceedings of the 30th Brazilian Symposium on Software Engineering, SBES '16, ACM,

New York, NY, USA, 2016, pp. 53–62. doi:10.1145/2973839.2973845.
URL `http://doi.acm.org/10.1145/2973839.2973845`

[5] M. Httermann, DevOps for developers, Apress, 2012.

[6] J. Allspaw, P. Hammond, 10+ deploys per day: Dev and ops cooperation at flickr, talk presented at Velocity: Web Performance and Operations Conference (2009).

[7] P. Labs, DevOps Research, D. Assessment, 2017 state of DevOps report, Tech. rep., retrieved May, 2018 from `https://puppet.com/resources/whitepaper/state-of-devops-report` (2018).

[8] L. Riungu-Kalliosaari, S. Mäkinen, L. E. Lwakatare, J. Tiihonen, T. Männistö, Devops adoption benefits and challenges in practice: A case study, in: P. Abrahamsson, A. Jedlitschka, A. Nguyen Duc, M. Felderer, S. Amasaki, T. Mikkonen (Eds.), Product-Focused Software Process Improvement, Springer International Publishing, Cham, 2016, pp. 590–597.

[9] F. Erich, C. Amrit, M. Daneva, Cooperation between information system development and operations: A literature review, in: Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM '14, ACM, New York, NY, USA, 2014, pp. 69:1–69:1. doi:10.1145/2652524.2652598.
URL `http://doi.acm.org/10.1145/2652524.2652598`

[10] F. M. A. Erich, C. Amrit, M. Daneva, A qualitative study of DevOps usage in practice, J. Softw. Evol. Process 29 (6) (2017) n/a–n/a. doi:10.1002/smr.1885.
URL `https://doi.org/10.1002/smr.1885`

[11] W. P. Luz, G. Pinto, R. Bonifácio, Building a collaborative culture: a grounded theory of well succeeded DevOps adoption in practice, in: Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM 2018, Oulu, Finland, October 11-12, 2018, 2018, pp. 6:1–6:10.

[12] B. Kromhout, Containers will not fix your broken culture (and other hard truths), Queue 15 (6) (2017) 50:46–50:56.

[13] D. I. K. Sjøberg, T. Dybå, B. C. D. Anda, J. E. Hannay, Building Theories in Software Engineering, Springer London, London, 2008, pp. 312–336.

[14] C. Wohlin, P. Runeson, M. Hst, M. C. Ohlsson, B. Regnell, A. Wessln, Experimentation in Software Engineering, Springer Publishing Company, Incorporated, 2012.

[15] B. G. Glaser, A. L. Strauss, The Discovery of Grounded Theory: Strategies for Qualitative Research, Observations (Chicago, Ill.), Aldine Publishing Company, 1967.

[16] G. Coleman, R. O'Connor, Using grounded theory to understand software process improvement: A study of irish software product companies, Information and Software Technology 49 (6) (2007) 654–667.

[17] K. Charmaz, Discovering chronic illness: using grounded theory, Social science & medicine 30 (11) (1990) 1161–1172.

[18] J. H. Barnsteiner, Using grounded theory in nursing, Journal of Advanced Nursing 40 (3) (2002) 370–370.

[19] S. A. Hutchinson, Education and grounded theory, Journal of Thought (1986) 50–68.

[20] G. Kenealy, Management research and grounded theory: A review of grounded theory building approach in organisational and management research, The Grounded Theory Review 7 (2) (2008) 95–117.

[21] R. Hoda, J. Noble, Becoming agile: A grounded theory of agile transitions in practice, in: Proceedings of the 39th International Conference on Software Engineering, ICSE '17, IEEE Press, Piscataway, NJ, USA, 2017, pp. 141–151. doi:10.1109/ICSE.2017.21.
URL `https://doi.org/10.1109/ICSE.2017.21`

[22] K.-J. Stol, P. Ralph, B. Fitzgerald, Grounded theory in software engineering research: A critical review and guidelines, in: Proceedings of the 38th International Conference on Software Engineering, ICSE '16, ACM, New York, NY, USA, 2016, pp. 120–131. doi:10.1145/2884781.2884833.
URL `http://doi.acm.org/10.1145/2884781.2884833`

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

[24] N. K. Denzin, Grounded theory and the politics of interpretation, The Sage handbook of grounded theory (2007) 454–471.
[25] S. Jantunen, D. C. Gause, Using a grounded theory approach for exploring software product management challenges, Journal of Systems and Software 95 (2014) 32–51.
[26] R. Hoda, J. Noble, S. Marshall, The impact of inadequate customer collaboration on self-organizing agile teams, Information and Software Technology 53 (5) (2011) 521–534.
[27] S. Adolph, P. Kruchten, W. Hall, Reconciling perspectives: A grounded theory of how people manage the process of software development, Journal of Systems and Software 85 (6) (2012) 1269–1286.
[28] S. Georgieva, G. Allan, Best practices in project management through a grounded theory lens., Electronic Journal of Business Research Methods 6 (1) (2008) 43–52.
[29] F. Shull, J. Singer, D. I. Sjøberg, Guide to advanced empirical software engineering, Springer, 2007.
[30] D. L. Morgan, Focus groups, Annual review of sociology 22 (1) (1996) 129–152.
[31] L. Lehtola, M. Kauppinen, S. Kujala, Requirements prioritization challenges in practice, in: F. Bomarius, H. Iida (Eds.), Product Focused Software Process Improvement, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 497–508.
[32] W. Luz, E. Agilar, M. C. de Oliveira, C. E. R. de Melo, G. Pinto, R. Bonifácio, An experience report on the adoption of microservices in three brazilian government institutions, in: Proceedings of the XXXII Brazilian Symposium on Software Engineering, SBES '18, 2018, pp. 32–41.
[33] J. Lewis, F. Martin, Microsservices, `http://martinfowler.com/articles/microservices.html`, accessed: 2018-05-22 (2014).
[34] R. Hoda, J. Noble, S. Marshall, Developing a grounded theory to explain the practices of self-organizing agile teams, Empirical Software Engineering 17 (6) (2012) 609–639.
[35] J. Willis, What DevOps means to me, retrieved from `https://blog.chef.io/2010/07/16/what-devops-means-to-me` (2010).
[36] G. Vergori, D. A. Tamburri, D. Perez-Palacin, R. Mirandola, Devops performance engineering: A quasi-ethnographical study, in: Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering Companion, ICPE '17 Companion, 2017, pp. 127–132.
[37] G. Kim, K. Behr, G. Spafford, The Phoenix Project: A Novel About IT, DevOps, and Helping Your Business Win, 1st Edition, IT Revolution Press, 2013.