How to organize DevOps' teams in customer firms? A comparative case analysis

Lamiae Benhayoun
Rabat Business School, International University of Rabat
lamiae.benhayoun@uir.ac.ma

Charmaine Carda
Institut Mines Télécom Business School
charmaine.carda@imt-bs.eu

Abstract

The issue of DevOps' team structures has been seldom addressed in prior literature. Some scholars underlined certain teams' characteristics and proposed structure taxonomies. However, they hardly considered the effect of the firm's context on a structure choice, by only focusing on project-level influences in few studies. To cover this gap, we propose an organizational model for DevOps' implementation within a consulting configuration. This setting is frequent in the current digital era and is particular as the partners differ in terms of digital maturity, cognition, and goals. We explored three cases in the public administration, telecommunications, and banking sectors. Data was collected through participant observation and semi-structured interviews and thematically analyzed with Nvivo. As results, we identify the key components of DevOps' teams and highlight their synergies. We also contribute to academic literature and managerial practice by raising customer firms' awareness on the contextual factors inducing variability on DevOps' team structure.

Keywords: DevOps; Agile methods; Software development; Team structure; Multiple case studies

1. Introduction

The increasing demand of customized IT solutions and the need of supplying software fast into the market becomes crucial to maintain competitive advantages (Sarah & Fakieh, 2020). IT teams must hence adapt new technologies and enroll a customer centric perspective. Currently, DevOps is the cutting-edge approach for software development emphasizing operational aspects and agile notions (Perera et al., 2017). It is an organizational and cultural transformation that aims for collaboration among all stakeholders involved in the development, deployment, and operation of software to deliver a high-quality product or service in the shortest possible time (Lopez-Fernandez et al., 2021). DevOps emerged from continuous deployment as an evolution of agile software development, informed by lean principles (Lwakatare et al., 2016). This background of DevOps suggests that small and frequent changes are performed with focused value to the end customer (Cook et al., 2012; Ebert et al., 2016). Thus, the software is always in a releasable state throughout its lifecycle, and is delivered in a cheaper and faster way (Zhu et al., 2016).

DevOps has been examined in literature from different angles. Several scholars underlined its benefits for instance reducing the development and deployment time and cost, enhancing the deployment rates and the stability of the project, and optimizing the Mean Time to Recover (Hüttermann, 2012). Other researchers explored the enablers of DevOps usually conveying culture, automation, measurement and sharing (Humble & Farley, 2010), while some studies addressed the barriers including the cognitive distance between the Dev and Ops' teams (Rütz, 2019) and the technical ineffectiveness of rollback methods used for automation (Kamuto & Langerman, 2017). Finally, very few studies examined the organization of DevOps' teams. On the one hand, some authors underlined certain characteristics of the DevOps' team such as platform teams (Bahadori & Vardanega, 2018), or the absence of a formal leader (Spiegel et al., 2021). On the other hand, others proposed some taxonomies of the team structure (e.g. Lopez-Fernandez et al., 2021; Shahin et al., 2017).

However, none of these studies efficiently guide organizations towards the appropriate DevOps' team structure as they hardly consider the contextual peculiarities of DevOps' implementation. The few existing research only took account of project-level properties such as individuals' roles and the project size and aim (Leite et al.,2020a; Shahin et al., 2017). Considering the effects of DevOps' context is crucial when organizing DevOps' teams in order to fit best to the organization's goals and to enable high cooperation and a shared responsibility between all functional areas. DevOps' context goes beyond project-level and incorporates the product set of an organization and its maturity to take the right lead on operational concerns (Skelton & Pais, 2019). In the present study, we cover this gap by proposing an organizational model for DevOps' implementation, linking the perspectives of client organizations and the consultant entity supporting DevOps' deployment.

Accordingly, we raise the following research question: How should DevOps' teams be organized in customer firms? Indeed, in today's increasingly turbulent environment, organizations aim to focus on their core expertise and therefore tend to outsource IT projects to consultancy teams who will accompany the induced change (Henningsson et al., 2016). However, the customer and consulting firms may have different strategic and cognitive goals (Lwakatare et al. 2019), as well as disparate levels of maturity with respect to the DevOps' framework (De Bayser et al., 2015). Also, DevOps requires an increased level of trust (Stray et al., 2018), which takes more time to establish between partnering organizations with different orientations than within the same organization (Bonfim et al., 2017).

To answer our research question, we explored three cases in the public administration, telecommunications, and banking sectors. Data was collected through participant observation and semi-structured interviews and thematically analyzed with Nvivo. We contribute to academic literature and managerial practice by guiding customer companies towards the key components to include in their teams and raising their awareness of the contextual circumstances that affect the components' roles.

2. Theoretical background

2.1. The DevOps' concept

DevOps is a set of practices and cultural values that has emerged in the software development industry. It emphasizes automation, virtualization, and new tools for collaboration between software development and operations (Humble & Farley, 2010). It uses specific principles of collaboration and automation to manage deployment environments and their configurations (Lwakatare et al., 2015). Collaboration in DevOps seeks to bridge the silos of software development and operations' functions, which exist as separate functions in most companies. Their split is mostly due to different goals and incentives that are owned by the two separate organizational units. Developers want to push changes into production as fast as possible, whereas operations' personnel want to keep production environment stable (Humble & Farley, 2010).

Throughout the development process, DevOps focuses on the communication holes between development and operations. Accordingly, information about usage of features in production and the performance of the system are communicated early to the development team. This helps to facilitate the establishment of continuous improvements to existing or to new products, not only in the web domain but also in the embedded systems' domain (Olsson & Bosch, 2013; Pérez et al., 2015).

Prior studies underlined several benefits of DevOps. First, it improves the collaboration between the development and operation teams, which reduces miscommunication (Diaz et al., 2019), enables the frequent integration of both teams' activities (Luz et al., 2019; Toh et al., 2019), and even enhances the team morale (Senapathi et al., 2018). Then, DevOps can improve software deployment speed from months to days (Lwakatare et al., 2019). It reduces the software process cycle time (Toh et al., 2019) and enables fast time to market (Diaz et al., 2019; 2021). Furthermore, it leads to better software quality thanks to the use of automation methods that support fast feedback loop, and continuous delivery (Lwakatare et al., 2019). In addition, DevOps can help reduce software downtime (Toh et al., 2019) by increasing the software ability to recover rapidly (Diaz et al., 2019; Erich et al., 2017). Finally, DevOps can decrease development cost, because of shorter development cycle and more frequent deployments (Diaz et al., 2021).

2.2. Factors affecting DevOps' implementation

Several researchers focused on the factors affecting the implementation of DevOps, either its enablers or barriers. Regarding the enablers, the CAMS model (culture, automation, measurement and sharing) usually represents the fundamental examined factors (Humble & Farley, 2010). Collaborative culture is an important building block as it supports social interactions necessary for software development, especially within flatten hierarchies (Banica et al., 2017). Automation focuses on the technical use of specific tools to automate the process from development to operation (Ebert et al., 2016). The measurement enabler means that all actions and processes must be mirrored in KPIs to measure the delivery capability and the implementation of a continuous improvement framework (Humble & Molesky, 2011). The sharing factor conveys that openness and transparency allow an effective cooperation between development and operation teams. It relies on suitable tools, culture, ideas, problems learned, and data (Humble & Farley, 2010).

As for the barriers to DevOps, literature emphasizes the human aspect, especially the divergence of views between the Dev and Ops' teams. The operation team relies on stability and reliability, whereby the development team looks for change and new innovative tools. Also, DevOps' adoption needs the right people in the right position (Rütz, 2019). We then note a technical barrier associated with the ineffectiveness of rollback methods used for automation and frequent deployments within a day (Kamuto & Langerman, 2017).

2.3. DevOps' team structure

The intricate and challenging nature of DevOps entails changes in the structure of teams, people mindset, and the sets of skills to capitalize optimally on this framework (Claps et al., 2015; Leppänen et al., 2015). In this respect, very few papers underlined some characteristics of the DevOps' teams. First, the latter is horizontal and can be formed by people exclusively dedicated to platforms or by people from each product team specialists in DevOps. This horizontal configuration helps decrease the cognitive load of the product teams and improve their productivity (Lopez Fernandez et al., 2021). Then, Bahadori and Vardanega (2018) stressed the role and importance of platform teams. Specifically, they discussed why product teams require infrastructure agility and how dynamic orchestration of infrastructure delivery may accelerate software delivery. In addition, a key characteristic of the structure is the high cooperation between the Dev and Ops' teams (Nybom et al., 2016). In this vein, Stray et al. (2018) underlined that mature DevOps' team structures are characterized by the absence of silos, the sharing of goals and of the ownership of the product they are building, internal trust, and limited dependency on external teams. Also, highly efficient team structures have a single formal leader. As the team evolves, it becomes self-organized, and the leadership is progressively transferred from the formal leader to the team (Spiegel et al., 2021).

Some of these studies focused on identifying the different organizational structures adopted to arrange development and infrastructure professionals (Lopez-Fernandez et al., 2021; Macarthy & Bass, 2020; Mann et al., 2019; Nybom et al., 2016; Shahin et al., 2017; Skelton & Pais, 2019). A pioneering work is that of Shahin et al. (2017) who identified four types of structures: separate Dev and Ops' teams with higher collaboration, separate Dev and Ops' teams with facilitator(s) in the middle, small Ops' team with more responsibilities for Dev team, and no visible Ops' team. Leite et al. (2020a; 2020b; 2021) collected data from 27 IT professionals and identified four organizational structures: siloed departments, classical DevOps, cross-functional teams, and platform teams. However, none of these studies provide organizations with guidance to implement DevOps considering their contextual peculiarities. Indeed, DevOps' scenarios can have sources of uncertainty which should be identified and addressed (Shahin et al., 2017). Proposing an organizational model of DevOps and ways to adjust according to the organization's characteristics would help to overcome uncertainties and better manage resources. The DevOps' context largely determines how businesspeople, developers, and operators should be organized and how DevOps' culture is adopted across the organization (Lopez-Fernandez et al., 2021).

In the present research, our ambition is to cover this gap by proposing an organizational model for DevOps' implementation, linking the perspectives of client organizations and the consultant supporting DevOps' deployment. In today's increasingly turbulent environment, organizations aim to focus on their core expertise and therefore tend to outsource IT projects to consultancy teams who will accompany the induced change (Henningsson et al., 2016). Firms seek to benefit from consultants' renowned strength in creating software applications and steering the subsequent organizational transformation (Tomo et al., 2021). The customer and the consultant teams need to work closely to enable fast and frequent delivery and deployment of new and changing features while ensuring quality and non-disruption of the production and deployment environments (Diaz et al., 2019). However, they may have different strategic and cognitive goals (Lwakatare et al. 2019), as well as disparate levels of maturity with respect to the DevOps' framework (De Bayser et al., 2015). Also, DevOps requires an increased level of trust (Stray et al., 2018), which takes more time to establish between partnering organizations with different orientations than within the same organization (Bonfim et al., 2017). The issue is magnified as the consulting and client firms rely on different tools and reward structures (McCarthy et al., 2015).

3. Research methodology

3.1. Data collection

3.1.1. Participant observation

We relied on multiple sources of qualitative data to reach the research purposes. First, one of the researchers held a position of an IT consultant for three firms in the public administration, telecommunications, and banking sectors. Therefore, it was appropriate to implement a participant observation approach within these case companies to gather relevant data. This method has long been used in Information Systems' studies and enables the researcher to act both an observer and a participant in some activity over time (Nandhakumar & Jones; 2002). We employed participant observation in the present research because, thanks to the consulting missions, "*the researcher shares as intimately as possible in the life and activities of the people in the observed setting*" (Genzuk, 2003, p.2).

We used fictitious names (FinGov, TelCo, BigBank) to preserve the anonymity of these organizations. The consulting missions were contracted with 'Consul', an international firm specializing in IT and organizational transformation. The participant observation focused among others on analytical workshops, planning meetings, release test sessions, sprint reviews, etc. The three observed cases were selected to cover a wide range of DevOps' implementation contexts. First, the companies belong to three industries that depicts diverse peculiarities likely to induce variance in DevOps' implementation. The sectors are characterized by disparate industry sizes as well as different intensities of technological evolution and competition (Bower & Christensen, 1995). Second, the cases differ in terms of their maturity with respect to the DevOps' framework and their governance mode of IT projects, which would result in variations regarding their DevOps' adoption perspectives (Lindner et al., 2016). These elements are detailed in the following paragraphs.

The mission in FinGov lasted for 18 months. FinGov is a public finance administration and was not previously familiar with Agile or DevOps practices. The structure was very hierarchical and vertical. The IT Department is made up of a central decision-making body in charge of steering, and of several regional bodies acting as internal outsourcers. These bodies are responsible for development, workstation support, system operation, user assistance, and include development teams, system operators, etc. With the help of a global mapping of its resources on the national territory, FinGov makes sure to distribute the activities between the different regions. The teams manage these activities in a V cycle project logic.

TelCo is a virtual operator, subsidiary of a historical operator and an insurer. The organization wants to quickly develop its IT services and eliminate the break between the 'Build' and the 'Run' induced by the management of the Run by the historical host of the insurer. The mission focused on this issue and lasted for 12 months. The Build defines operations to propose new solutions, while the Run specifies the production practices to deliver a good or service with a goal of quality and continuous improvement. The Build works in agile with current technologies, while the Run is very traditional with infrastructures that are not cloudy. The application team is limited to an application development perimeter, and the internal operators are solely limited to the technical integration perimeter.

BigBank is one of the world leaders in the banking sector. This company has been accelerating its transformation, in particular with a move to the Cloud, which is now managed internally. Consul has been implementing DevOps' practices in BigBank for four years incrementally through various missions. In order to push the DevOps' logic to its climax, the researcher-consultant was involved for 6 months in the setting up of site reliability engineering profiles, i.e. development engineers at the service of operations and infrastructure. The application teams include developers, Product Owners, and Scrum Masters organized according to the SAFe method. All the tools and procedures of the chain are automated at the scale of a service.

3.1.2. Semi-structured interviews

Participant observation has the distinctive power of describing complex social interactions and cognitive and cultural aspects over time, but has some limitations: it represents a single perspective, requires subjective introspection (Pronin, 2007), and may suffer from the effect of observer-expectancy, where the observer's presence affects the attitude of other participants (Ko, 2017). To mitigate these limitations, we triangulated the data resulting from the observation with other data sources (Aktinson & Hammersley, 1998). In this respect, we performed semi-structured interviews with key actors of the customers' teams as explained in Table 1. Evidence was also gathered through informal conversations with the Dev and Ops teams' participants on how they perceived the project unfolding and its outcomes. In addition, we collected valuable complementary documents such as training guides, meeting minutes, annual reports, performance dashboards, organizational frameworks, formal procedures, and information notices.

Table 1. Characteristics of the interviews

Case	Number and duration	Types of interviewees	Mode
FinGov	7 interviews ≈130 minutes	• IT manager • Project manager • Software developer • 2 System developers • Operational manager • Network administrator	Face-to-face
TelCo	5 interviews ≈100 minutes	• Software developer • DevOps' engineer • Network engineer • Chief Technology Officer • Cloud architect	Face-to-face, Skype & phone
BigBank	5 interviews ≈75 minutes	• Quality assurance expert • Software developer • System developer • DevOps' expert • Cloud architect	MS Teams and phone

To prepare for data collection through interviews, we performed an in-depth literature review of the extant DevOps' team structures in prior research and accordingly devised an interview guide. The guide was composed of three distinct parts. The first part was dedicated to understanding the interviewee's role in the firm as well as his educational and professional background. The second part addressed the initial organization of the customer team before the transformation induced by DevOps. Finally, the third part focused on the characteristics of the new organization following DevOps' implementation during the consultant's missions. This interview guide was tested with three researchers and four practitioners before its deployment within the cases.

17 interviews were performed within the three cases and generated 300 minutes (5 hours) of recordings, which were transcribed and grammatically sub-divided to enable their thematic analysis with NVivo. After an interview, we used the emerging results to enrich those of the participant observation as described in the next section. Saturation was reached by the 14th interview.

3.2. Data analysis

We relied on NVivo to analyze the data collected within the case companies. This approach combined established methodologies for grounded theory building (Gioia et al., 2013) and multiple case analysis (Miles et al., 2013). It involved travelling back and forth between the data and the emerging structure of theoretical arguments (Locke, 2007) as explained below:

On the one hand, we followed the recommendations of Gioia et al. (2013) to identify in our corpus the components of the DevOps' team structure to include in our model. We first proceeded to a 1st-order inductive coding using informant-centric terms, to identify distinctive concepts from the cases. These concepts refer to the team components and roles, as reported by the interviewees or as observed by the researcher. As the research progressed, we started seeking similarities and differences among the many emergent concepts by using axial coding, which enabled gathering similar concepts, that we labeled using the informant terms.

Then, based on the identified concepts, we performed a 2nd-order analysis using researcher-centric terms (Gioia et al., 2013) to explore whether the relationships among the emerging concepts suggest higher order themes to describe and explain the phenomena we are observing. These 2nd-order themes represent more robust theoretical descriptions of the team components derived from the 1st-order concepts. Once a workable set of concepts and themes was in hand, we analyzed their underlying nature to investigate if it was possible to distill the emergent 2nd-order themes even further into 2nd-order aggregate dimensions as recommended by Gioia et al. (2013).

On the other hand, NVivo enabled us to conduct a comparative analysis of the three cases. This analysis highlighted regularities and differences between the cases throughout the entire corpus (Eisenhardt, 1989). We identified the recurrent team components and uncovered several contextual factors that induce variance regarding the presence of the model's components within the DevOps' team structure.

To ensure intercoder reliability and comfort the robustness of our findings, the empirical corpus was coded twice by two of the authors. This double analysis resulted in an 85% agreement between both coders, considered sufficient for intercoder reliability (Krippendorff 2004).

4. Results

Appendix 1 (https://bit.ly/3VddR5w) recalls the contextual differences between the three cases and synthesizes the key findings regarding the DevOps' team structures and the change process in each firm. The results are detailed in the next paragraphs and illustrated with verbatim based on the participants' feedback. The consultant was involved in the three DevOps' teams but possessed a temporary position. He provided a supportive role since he accompanied the teams in the design of software applications and was responsible for monitoring the ensuing organizational transformation and the transition to full DevOps.

4.1. FinGov case

4.1.1. Description of the resulting model

Consul led an acculturation and implementation of Agile and DevOps' practices simultaneously. The main objective of agility was to configure the application teams into Squads made up of 5 to 12 members. *"We wanted to have multi-skilled teams responsible for the life cycle of all or part of an application product including design, implementation and integration" (FinGov).*

Consul brought out two configurations of application teams following discussions with the client: a minimum base and a complementary team. FinGov was looking for a configuration allowing the integration of Ops within the application teams. *"This responds to a desire to mature the application teams regarding application run issues." (FinGov).*
For the creation of these pilot teams, *"we have selected, with the help of Consul, different roles in the regional IT bodies in order to make them experiment within application teams of the central body" (FinGov).*
The minimum base is a team including the skills deemed essential to the creation of a squad with the three key roles of a Scrum team: The Product Owner is the main point of contact for the business lines within the Squad and is responsible for creating and managing the Backlog and for delivering the final product. The Scrum Master is the facilitator, the team coordinator, and the supporter of the Agile method within the Squad. The development team carries the complete realization of the product (analysis, design, documentation, coding, technical tests, etc.) and ensures quality.
The complementary team enables to enrich the minimum base with three other roles: The Business Analyst supports the Product Owner, is involved in the functional design of the product, and oversees functional testing. The architect is responsible for the technical and functional architectural choices. He analyzes the impact of product changes. He ensures the quality, security, and consistency of the code and the implementation of efficient development practices. He supports the squad in carrying out technical tests and feeds the work of the Ops. The System Operator guarantees the availability and maintenance of environments for the squad, from development to production. He supports integration activities, usability testing, and commissioning and is responsible for setting up a continuous deployment chain.

As for the DevOps' part, Consul aimed to prepare the teams for managing applications in the internal cloud. The consultant accompanied the pilot projects for the definition of the architecture, the deployment of the infrastructure as Code, the configuration of the environments, and the implementation of the automated deployment and delivery chain. *"The precepts of agile implemented alongside DevOps have facilitated the management of applications. The Squad is responsible for creating and maintaining the environments it needs thanks to the Ops that have been integrated into it. These same actors intervene in the Build and Run." (FinGov).*

The teams tended increasingly towards the logic of 'product' squad. This logic implies that the squad has an end-to-end responsibility, from design to production. Development rates are incremental, with the delivery of an MVP (Minimum Viable Product) to the user.

4.1.2. Circumstances of the transformation

FinGov aimed at accelerating its digital transformation with the integration of the cloud. This was an opportunity to experiment with a new organization inspired by DevOps in order to create and exploit its IT applications. *"Using an Agile/DevOps framework common to all application teams seems relevant to us insofar as there is a real need to standardize practices between teams and within them" (FinGov).*
The experimentation took place within 8 projects, that volunteered to put new approaches into practices. At the start of the initiative, *"Consul helped each project manager to think about what resources we need, how to divide the activities, etc." (FinGov).*
Following this DevOps' experimentation within FinGov, no development project has resulted in the complete construction of an automated delivery chain. Also, no construction of production environments or of any technical production has been triggered.

Even though this initiative was supported by volunteer pilot teams, Consul encountered specific resistance regarding the resulting organizational change.
On the one hand, some members expressed the fear that the integration of new practices and a new organization could call into question the existence of their work. *"In the initial organization, the production pilots are in charge of studying the production release files to make 'go' and 'no go' decisions. If DevOps' practices were to be generalized, they would induce automated production releases every day. Production pilots could therefore see their profession disappear or being transformed." (FinGov).*
On the other hand, a DevOps' acculturation across all IT bodies was highly challenging. Particularly, *"we suffer from a tense social climate, due to approximately 1,000 job cuts per year. It is therefore not easy to propose big bang reorganizations and acculturations within the entire structure". (FinGov).*
The client, hence, chose to specialize only four of its IT regional bodies in DevOps. The ambition, by the end of 2022, is to have 15% of projects in Agile DevOps.

4.2. TelCo case

4.2.1. Description of the resulting model

Consul supported TelCo in its transformation initiative, more particularly in the definition of its Cloud and Delivery strategy, in the structuring of its new Cloud and DevOps' platform, in the management of its Cloud migration and transformation, and in the definition of a TOM (Target Operating Model). It was agreed to implement a TOM composed of several entities.

The teams are organized according to a SAFe mode. A Program Increment Planning is organized every two months and the Sprints last for two weeks. During these plannings, the application teams, the platform team, and the business representatives were necessarily present to prioritize the tasks at the right level. The executive committee could also attend.
We hereby explain the components of the TOM, which enabled to make the most of the Cloud.

The delivery layer is the higher organizational entity in the model and includes:
•Feature Teams: They oversee software delivery and reduce inter-team dependencies. Indeed, *"thanks to the multidisciplinary of the team, the members can develop a large number of features almost independently" (TelCo).*
These teams are responsible for the technical and functional specifications, as well as the design, development, and deployment of the applications.
•Transverse team: Responsible for agile methods, its role is to carry out coaching and support as needed. It thus manages continuity in case of agile implementation in a unit and steers, as well, the communities of practice.

Below the delivery layer, we find the cloud platform management layer composed of:
•Platform Team: This is the essential element of the transformation. Its role is to manage the infrastructure services offered by the cloud, for instance IaaS and PaaS, and to make them available to the application development teams. *"The Cloud and its management add a layer of complexity to the IS architecture by introducing new concepts such as containers or container orchestrators. This greatly affects traditional administration practices". (TelCo).*
For its internal management, *"the team is in a 'you build it, you run it' mode, i.e. it manages the design, the Build and the Run of everything it implements on the platform." (TelCo).*
Regarding the management of DevOps' tools, the Platform Team provides infrastructure services in API mode that can be used on a turnkey basis and is responsible for provisioning the Infrastructure as Code. *"If a feature team wishes to use IaC services, the Platform Team makes them available in API format. Sometimes the service can be provided turnkey" (TelCo).*
The Platform team configures models, creates monitoring tools for the application teams and is in charge of security activities on the Run. *"It provides a Kubernetes cluster allowing application teams, in the continuous delivery chain, to provision a specific container with its application." (TelCo).*
•Operational Security Center: Responsible for responding to security incidents in an end-to-end basis. It performs audit procedures to anticipate incidents.

Right beneath, the service offerings' layer includes:
•Cloud services: They incorporate the infrastructure models available with AWS (IaaS, PaaS, CaaS, SaaS) and access management and certification.
•Service workplace: It brings together the services in charge of the employees' work environments.
•Legacy entity: It relates to the historical information system and includes the residual infrastructures still operational after each transformation increment.

Next to these operational units, the model includes two structural bodies:
•Design Authority: It ensures technological consistency across the entire IT department. *"It's a cross-functional governance team, a kind of community of architects making global decisions. Some members of the application teams can simultaneously be part of it" (TelCo).*
This entity defines the enterprise architecture strategy and models and manages the aspects of governance and compliance risk.
•Transversal functions: They support the activity of the IT administration at the financial level, innovation, etc. These functions include, among others, the management of costs and suppliers, and the Helpdesk.

4.2.2. Circumstances of the transformation

The client wished to quickly develop its IT services and eliminate the break between the Build and the Run induced by the management of the Run by the insurer's historical host. To put an end to this rupture, *"we have decided to move to internal management of the AWS Cloud and strengthen DevOps' practices within the teams". (TelCo).*

As explained in the TOM introduced in the previous paragraph, the platform team was the central element of the transformation carried out within TelCo. It brings a new paradigm, consisting of the transfer of the application Run to the features teams. *"The creation of this Platform Team was quite ambitious insofar as we had to mature on the subject" (TelCo).*
Accordingly, the change occurred gradually. *"The transformation of the teams as well as the transition to responsibility on the Run took place in two stages, since the entire DevOps' transformation was correlated with the migration to the cloud, and thus to the newly created AWS platform. The transformation was implicitly linked to a sort of transfer of certain activities which were historically managed by the insurer's host and taken over by TelCo." (TelCo).*
•A first post-migration phase during the first half year, with the effective transition to the Cloud: This phase was characterized by the establishment of a Platform Team but also an AppOps' team. The latter is in charge of recuperating in Run the migrated applications on the AWS cloud. *"The AppOps' Team takes over application outsourcing activities that were traditionally operated by the insurer's host and thus takes over part of the integration activities." (TelCo).*
•A second phase at the end of the second half year with only the feature teams: This is the transition phase to full DevOps. It is characterized by the integration of AppOps' profiles within the feature teams, which are each in charge of their application products. The AppOps' activities absorbed by the feature teams concern integration, operation, application supervision, and monitoring to drive the performance of applications under development. Indeed, for the resulting feature teams, *"what they were designing, they were also overseeing it at the Run level from end-to-end." (TelCo).*
The members of the final feature teams are: Product Owner, Scrum Master, Tech Lead, AppOps whose role is to infuse good operating practices in the teams, developers, and a Security Champion.
Thanks to this two-step transformation, *"the AppOps have participated in the acculturation of the developers to the*

applicative exploitation. By absorbing the members of the AppOps' team, the features teams have been able to increase their skills in activities traditionally of an Ops' nature. The AppOps who integrate these teams no longer do traditional Ops' activities but rather operate in a DevOps' ecosystem, with the use of the continuous delivery chain and Infrastructure as Code." (TelCo).

Each feature team has complete autonomy over the automation and industrialization it wishes to implement. *"However, there are good development and management practices that these teams must respect to guarantee the usability of the applications. There are structuring choices that must be made on the tools and the selection of a solution according to the needs." (TelCo).*

In short, for this project, the shaping of AppOps and Platform teams was the most significant element of transformation. *"It took the longest time because it was not trivial to develop the right reflexes within the feature teams which are now in charge of the Application Run." (TelCo).*

4.3. BigBank case

4.3.1. Description of the resulting model

BigBank has been accelerating its transformation with a move towards the Cloud. Consul carried out the implementation of DevOps and agile practices incrementally over four years through various missions. The DevOps' model already deployed at BigBank consists of three layers:

•The upper layer concerns professional services and support for the transformation of business applications. It is structured into sub-departments, each focused on an element of activity of BigBank (investment banking, retail banking, etc.). Each sub-department has its own feature team responsible for the design, development, and delivery of applications.

•The second layer focuses on managing the cloud platform and its applications. It includes two units, one specializing in the cloud platform and the other in operations management.

•The last layer aims to design, maintain, and evolve service offerings. This layer groups the cloud infrastructure models and a workplace service.

BigBank decides to set up SRE (Site Reliability Engineering) profiles within the cloud platform management entity as well as in the application teams, i.e. development engineers at the service of operations. *"The challenge of SREs is to allow BigBank to position itself on complex products and to take charge of some of the Run incidents to improve the reliability of services". (BigBank).*

Consul, after consultation with BigBank, decided to disseminate SRE practices as follows:

•For the feature teams, transition their Ops to an SRE logic, by strengthening their DevOps' practices with the peculiarities of an SRE profile. This ultimately strengthens the accountability of Feature Teams on reliability issues.

•For the platform management entity, set up a new team whose role is to centrally manage reliability on public and private cloud platforms. The SREs deployed in the application teams would relay this team within the business departments.

These SRE profiles must know how to develop, design, and should have good knowledge of security and system administration. They act in several ways:

•In general, within organizations, when there is an incident, whether at the level of the customer who reports the incident or a monitoring tool, it arrives on a frontline. *"Within BigBank, there is no frontline on Cloud services. The interest is then to position in the management entity of the cloud platform an SRE team which serves as an entry point for incidents. For any incoming incident, the responders industrialize to solve the problem and prevent it from happening again. This enables moving to a higher level of reliability.". (BigBank).*

•Beyond incident handling, the SRE profiles are responsible for defining product operability standards.

•They automate Run tasks with Infrastructure as Code.

4.3.2. Circumstances of the transformation

The nerve center of the transformation in this case is the management of incidents and complex products, with the desire to move to a higher level of reliability. *"BigBank, due to its activity, has to process a large volume of operations and therefore incidents. It is hence relevant to want to move to the highest levels of reliability." (BigBank).*

The SRE is not to be considered as a replacement for a DevOps' operating mode characterized with a strong collaboration of Dev and Ops. *"The SRE for us is rather an extension of DevOps, even a crystallization on operational management subjects requiring greater expertise." (BigBank).*

The SRE appears as an embodiment of the DevOps' philosophy within a job description. These profiles support the application teams in their respect for the reliability of the developments, by achieving an increase in the skills of the Ops on this subject. *"This proposed tandem organization then makes it possible to strengthen the anchoring and dissemination of DevOps' practices within the information systems department." (BigBank).*

5. Discussion

This empirical study uncovered how DevOps' teams are organized in customer firms. Our answer to the research question provides a double contribution to literature. We first propose a model highlighting the components of DevOps' team structure for client firms. Second, we identify the determinants of DevOps's implementation in customer teams and describe their effects. These points are summarized in the DevOps' operating model in Appendix 2 (https://bit.ly/3VddR5w) and are discussed in the following paragraphs.

5.1. A model for DevOps' implementation in customer organizations

This study describes the components of DevOps' teams in customer firms. As such, we add to the extant scarce literature that shed light on some elements within DevOps' teams (e.g., Bahadori & Vardanega, 2018; Lopez Fernandez et al., 2021) and proposed structure taxonomies (e.g., Macarthy & Bass, 2020; Shahin et al., 2017; Leite et al., 2021). Our research stands out by underlining the fundamental components that must exist within any team and the complementary elements that are adapted to the DevOps' implementation context. The two types of bodies characterize a delivery layer, a platform layer, and a service offerings' layer:

For delivery, in line with prior studies (Ebert et al., 2016; Marnewick & Langerman, 2020), our results demonstrate that the key entity is the feature teams responsible for the technical and functional specifications, as well as the design, development, and deployment of the applications. Each team is organized in a Squad with multi-skilled members who oversee the life cycle of all or part of an application product. To allow such functioning, the feature teams must include Ops' profiles in charge of creating and maintaining production environments, hence enabling the team to take full responsibility of Build and of Run issues as well. These AppOps' profiles participate in the acculturation of the developers to the applicative exploitation. Our results additionally highlight complementary roles of SRE (Site Reliability Engineering) profiles within the feature teams that were hardly underlined in previous studies. These development engineers are at the service of operations (Bertolino et al., 2020). They define product operability standards and automate Run operations. Finally, for the delivery layer, our study unveils three entities extending the roles of the feature teams. These are a complementary Scrum team composed of a business analyst, architect, and system operator, an agile transverse team guarantor of agile methods, and a temporary AppOps' team in charge of recuperating in Run the migrated applications during the initial phase of DevOps' implementation in a client firm.

For the platform management layer, the vital entity as stressed out in several literature works is the platform team made up of specialists whose role is to decrease the cognitive load of product teams (Lopez Fernandez et al., 2021). To efficiently manage the infrastructure and makes it available to the feature development teams, the present research revealed that the platform team manages the design, the Build, and the Run of everything it implements on the platform. It provides infrastructure services in API mode that might be used on a turnkey basis and is responsible for provisioning the Infrastructure as Code. Our findings also shed light on the complementary role of SRE in this platform team, whose responsibility is to process incoming incidents and improve the levels of service reliability. Finally, for this layer, the only element supplemental to the platform team is the operations' management center that handles security services.

For the service offerings' layer, it necessarily includes Cloud services that administer the infrastructure models available within the cloud and the access management (Breiter et al., 2014). It might, alternatively, involve a service workplace and a legacy entity relating to the historical information system. Next to these operational units composing the three layers, the resulting model includes a transversal unit supporting the activity of the IT administration for example in terms of cost and vendor management (Hering, 2018), and a design authority, which has barely been evoked in prior studies. It is a cross-functional governance team made up of architects who decide on the enterprise architecture strategy and models.

5.2. Factors affecting the DevOps' model

We provide an important extension to literature on DevOps that examined the factors affecting the team structure but focused either on deployment enablers or barriers. Indeed, we identify the determinants of DevOps's implementation in customer teams and describe their effects. Determinants are contingency factors whose variations are followed systematically by variations in an outcome of interest. They represent contextual elements not controlled by an organization (Bauman et al., 2002). Considering these determinants is crucial to guide firms toward a team structure that fits their contexts. However, the very few typologies of DevOps' teams proposed in prior studies were not based on contextual factors and their authors did not explain how the firm's context would affect the suitability of a structure over another. For example, Shahin et al. (2017) proposed four types of structures, but the authors' categorization was based on the level of collaboration between Dev and Ops and their corresponding degrees of responsibilities. Both factors relate rather to controlled behavioral aspects than to contextual traits of the firm. We detail hereafter the determinants that emerged from our qualitative analysis and explain their potential effects on the model.

The first factor is related to the level of maturity regarding Agile. A lack of maturity requires the establishment of a Scrum configuration parallel to DevOp's implementation. This result provides support to the study of Lwakatare et al. (2016) who stated that agile is a prerequisite of DevOps. The second factor concerns the level of the customer firm's maturity with respect to DevOps, which impacts the deployment duration and complexity. The existence of an efficient DevOps' framework in the firm makes the change more at the margin through the implementation of supporting SRE profiles. In the opposite case, it is necessary to implement not only the key components proposed within our model (Feature teams, platform team, cloud services), but also additional entities to help the vital model bodies operate efficiently. We noted, for example, that the firms with a low DevOps' maturity needed to put in place temporary AppOps' teams to assist the feature teams for delivery, and an operations' management center to support the platform team through the handling of security issues. These findings are aligned with Adriano (2021) who emphasized that prior knowledge of DevOps can affect the course of the implementation. We add to this study by showing that not only prior knowledge, but also prior implementation of DevOps even partially in the organization is likely to induce such variance.

The third factor is associated with the perimeter of DevOps' implementation. When the transformation is experimental and focuses on a limited number of feature teams, it does not imply the compulsory establishment of complementary entities. If the change is definitive and concerns a much larger perimeter, not only the key model's components are deployed, but also supplementary even temporary entities such as AppOps' teams for delivery that would help the feature teams mature incrementally and then disappear. Through these results, we support previous studies on IT development stream that emphasized the determining impact of the project perimeter on the deployment plan (Guérineau et al., 2018). We particularly unveil this effect for DevOps' implementation within customer teams.

The last divergence factor among the three deployed models was the social climate in the client company. When the climate is tense, for example due to substantial job cuts, DevOps' deployment depicts some peculiarities: To foster the success of the transformation, DevOps' implementation in the case of an insecure social climate only focuses on the vital model's elements and concerns a limited part in the firm at the beginning of the transformation. Also, involving a consulting entity is particularly helpful to infuse best practices and monitor the change in a neutral manner. To the best of our knowledge, no prior study underlined the determining effect of the social climate regarding DevOps' team structure.

6. Conclusion

Despite its contributions, the present research has some limitations that pave the path to future empirical studies. First, we focused on three cases through interviews and participant observation. Even if the cases presented elements of diversity, they could limit the generalization of our results to contexts different from those addressed in the research. Further empirical studies could evaluate the extent to which our model is applicable or could be amended or enriched in other types of contexts. The second limitation concerns the effective measurement of the transformation impact. Indeed, the evaluation of the benefits was based on the interpretations and opinions of certain key members of the teams. A quantitative study with KPIs deployed on a larger scale in the organizations would help accurately measure the effect on performance of DevOps' implementation. This measurement could be carried out at several spaced moments to effectively pilot the DevOps' model and adjust it if necessary.

7. References

Adriano, D. M. (2021). DevOps and information technology service management: A problem management case study (Doctoral dissertation).

Aktinson, P., & Hammersley, M. (1998). Ethnography and participant observation. Strategies of Qualitative Inquiry. Thousand Oaks: Sage, 248-261.

Bahadori, K., & Vardanega, T. (2018). DevOps meets dynamic orchestration. In International Workshop on Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment (pp. 142-154). Springer, Cham.

Banica, L., Radulescu, M., Rosca, D., & Hagiu, A. (2017). Is DevOps another project management methodology?. Informatica Economica, 21(3), 39.

Bauman, A. E., Sallis, J. F., Dzewaltowski, D. A., & Owen, N. (2002). Toward a better understanding of the influences on physical activity: the role of determinants, correlates, causal variables, mediators, moderators, and confounders. American journal of preventive medicine, 23(2), 5-14.

Bertolino, A., Angelis, G. D., Guerriero, A., Miranda, B., Pietrantuono, R., & Russo, S. (2020). DevOpRET: Continuous reliability testing in DevOps. Journal of Software: Evolution and Process, e2298.

Bonfim, L. R., Segatto, A. P., & Takahashi, A. R. W. (2017). The structural, relational and cognitive dimensions of social capital on innovation and technology in interorganizational and intraorganizational settings.

Bower, J. L., & Christensen, C. M. (1995). Disruptive technologies: catching the wave. Harvard Business Review.

Breiter, G., Behrendt, M., Gupta, M., Moser, S. D., Schulze, R., Sippli, I., & Spatzier, T. (2014). Software defined environments based on TOSCA in IBM cloud implementations. IBM Journal of Research and Development, 58(2/3), 9-1.

Claps, G. G., Svensson, R. B., & Aurum, A. (2015). On the journey to continuous deployment: Technical and social challenges along the way. Information and Software technology, 57, 21-31.

Cook, N., Milojicic, D., & Talwar, V. (2012). Cloud management. Journal of Internet Services and Applications, 3(1), 67-75.

De Bayser, M., Azevedo, L. G., & Cerqueira, R. (2015, May). ResearchOps: The case for DevOps in scientific applications. In 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM) (pp. 1398-1404). IEEE.

Diaz, J., Almaraz, R., Pérez, J., & Garbajosa, J. (2019). DevOps in practice. In Proceedings of the 19th International Conference on Agile Software Development Companion-XP (Vol. 18).

Diaz, J., López-Fernández, D., Pérez, J., & González-Prieto, Á. (2021). Why are many businesses instilling a DevOps culture into their organization?. Empirical Software Engineering, 26(2), 1-50.

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

Eisenhardt, K. M. (1989). Building theories from case study research. Academy of management review, 14(4), 532-550.

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

Genzuk, M. (2003). A synthesis of ethnographic research. Occasional Papers Series. Center for Multilingual, Multicultural Research (Eds.). Center for Multilingual, Multicultural Research, Rossier School of Education, University of Southern California. Los Angeles, 1-10.

Gioia, D. A., Corley, K. G., & Hamilton, A. L. (2013). Seeking qualitative rigor in inductive research: Notes on the Gioia methodology. Organizational research methods, 16(1), 15-31.

Guérineau, B., Rivest, L., Bricogne, M., Durupt, A., & Eynard, B. (2018). Towards a design-method selection framework for multidisciplinary product development. In 15th International Design Conference (pp. 2879-2890).

Henningsson, S., & Øhrgaard, C. (2016). IT Consultants in Acquisition IT integration. Business & Information Systems Engineering, 58(3), 193-212.

Hering, M. (2018). DevOps for the Modern Enterprise: Winning Practices to Transform Legacy IT Organizations. IT Revolution.

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

Humble, J., & Molesky, J. (2011). Why enterprises must adopt devops to enable continuous delivery. Cutter IT Journal, 24(8), 6.

Hüttermann, M. (2012). Building blocks of devops. In DevOps for Developers (pp. 33-47). Apress, Berkeley, CA.

Kamuto, M. B., & Langerman, J. J. (2017). Factors inhibiting the adoption of DevOps in large organisations: South African context. In 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT) (pp. 48-51). IEEE.

Ko, A. J. (2017). A three-year participant observation of software startup software evolution. In 2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP) (pp. 3-12). IEEE.

Krippendorff, K. H. (2004). Content analysis: an introduction to its methodology. 2nd edition. Sage Publications, Thousand Oaks, California.

Leite, L., Kon, F., Pinto, G., & Meirelles, P. (2020a). Platform teams: An organizational structure for continuous delivery. In Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops (pp. 505-511).

Leite, L., Kon, F., Pinto, G., & Meirelles, P. (2020b). Building a theory of software teams organization in a continuous delivery context. In 2020 IEEE/ACM 42nd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion) (pp. 296-297). IEEE.

Leite, L., Pinto, G., Kon, F., & Meirelles, P. (2021). The organization of software teams in the quest for continuous delivery: A grounded theory approach. Information and Software Technology, 139, 106672.

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

Lindner, R., Daimer, S., Beckert, B., Heyen, N., Koehler, J., Teufel, B., ... & Wydra, S. (2016). Addressing directionality: Orientation failure and the systems of innovation heuristic. Towards reflexive governance (No. 52). Fraunhofer ISI Discussion Papers-Innovation Systems and Policy Analysis.

Locke, K. (2007). Rational control and irrational free-play: Dual-thinking modes as necessary tension in grounded theorizing. The SAGE handbook of grounded theory, 565-579.

López-Fernández, D., Diaz, J., Garcia-Martin, J., Pérez, J., & Gonzalez-Prieto, A. (2021). DevOps Team Structures: Characterization and Implications. IEEE Transactions on Software Engineering.

Luz, W. P., Pinto, G., & Bonifácio, R. (2019). Adopting DevOps in the real world: A theory, a model, and a case study. Journal of Systems and Software, 157, 110384.

Lwakatare, L. E., Kilamo, T., Karvonen, T., Sauvola, T., Heikkilä, V., Itkonen, J., ... & Lassenius, C. (2019). DevOps in practice: A multiple case study of five companies. Information and Software Technology, 114, 217-230.

Lwakatare, L. E., Kuvaja, P., & Oivo, M. (2015). Dimensions of devops. In International conference on agile software development (pp. 212-217). Springer, Cham.

Lwakatare, L. E., Kuvaja, P., & Oivo, M. (2016). Relationship of DevOps to agile, lean and continuous deployment. In International conference on product-focused software process improvement (pp. 399-415). Springer, Cham.

Macarthy, R. W., & Bass, J. M. (2020). An empirical taxonomy of DevOps in practice. In 2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA) (pp. 221-228). IEEE.

Mann, A., Stahnke, M., Brown, A., & Kersten, N. (2019). State of DevOps Report. Puppet & Splunk.

Marnewick, C., & Langerman, J. (2020). DevOps and Organizational Performance: The Fallacy of Chasing Maturity. IEEE Software, 38(5), 48-55.

McCarthy, M. A., Herger, L. M., Khan, S. M., & Belgodere, B. M. (2015, June). Composable DevOps: automated ontology based DevOps maturity analysis. In 2015 IEEE international conference on services computing (pp. 600-607). IEEE.

Miles, M.B., Huberman, A.M. and Saldana, J (2013). Qualitative data analysis. Sage.

Nandhakumar, J., & Jones, M. (2002). Development gain? Participant observation in interpretive management information systems research. Qualitative Research, 2(3), 323-341.

Nybom, K., Smeds, J., & Porres, I. (2016). On the impact of mixing responsibilities between devs and ops. In International Conference on Agile Software Development (pp. 131-143). Springer, Cham.

Olsson, H, H., & Bosch, J. (2013,). Towards data-driven product development: A multiple case study on post-deployment data usage in software-intensive embedded systems. In International Conference on Lean Enterprise Software and Systems (pp. 152-164). Springer, Berlin, Heidelberg.

Perera, P., Silva, R., & Perera, I. (2017). Improve software quality through practicing DevOps. In 2017 Seventeenth International Conference on Advances in ICT for Emerging Regions (ICTer) (pp. 1-6). IEEE.

Pérez, J. F., Wang, W., & Casale, G. (2015). Towards a devops approach for software quality engineering. In Proceedings of the 2015 Workshop on Challenges in Performance Methods for Software Development (pp. 5-10).

Pronin, E. (2007). Perception and misperception of bias in human judgment. Trends in cognitive sciences, 11(1), 37-43.

Rütz, M. (2019). Devops: A systematic literature review. no. August, 23, 25.

Sarah, A. L., & Fakieh, B. (2020). How DevOps Practices Support Digital Transformation. International Journal of Advanced Trends in Computer Science and Engineering, 9(3), May – June 2020, 27, 9(3).

Senapathi, M., Buchan, J., & Osman, H. (2018). DevOps capabilities, practices, and challenges: Insights from a case study. In Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018 (pp. 57-67).

Shahin, M., Zahedi, M., Babar, M. A., & Zhu, L. (2017). Adopting continuous delivery and deployment: Impacts on team structures, collaboration and responsibilities. In Proceedings of the 21st international conference on evaluation and assessment in software engineering (pp. 384-393).

Skelton, M., & Pais, M. (2019). Team topologies: organizing business and technology teams for fast flow. It Revolution.

Spiegler, S. V., Heinecke, C., & Wagner, S. (2021). An empirical study on changing leadership in agile teams. Empirical Software Engineering, 26(3), 1-35.

Stray, V., Moe, N. B., & Hoda, R. (2018). Autonomous agile teams: challenges and future directions for research. In Proceedings of the 19th international conference on agile software development: companion (pp. 1-5).

Toh, M. Z., Sahibuddin, S., & Mahrin, M. N. R. (2019). Adoption issues in DevOps from the perspective of continuous delivery pipeline. In Proceedings of the 2019 8th International Conference on Software and Computer Applications (pp. 173-177).

Tomo, A., Mangia, G., & Canonico, P. (2021). Innovating processes and processing innovation: strategic approach to innovation in accounting firms. Journal of Economic and Administrative Sciences.

Treitlier, I. (2014). Backyard ethnography: Defamiliarize the familiar to transform business. International Journal of Business Anthropology, 5(1), 93-105.

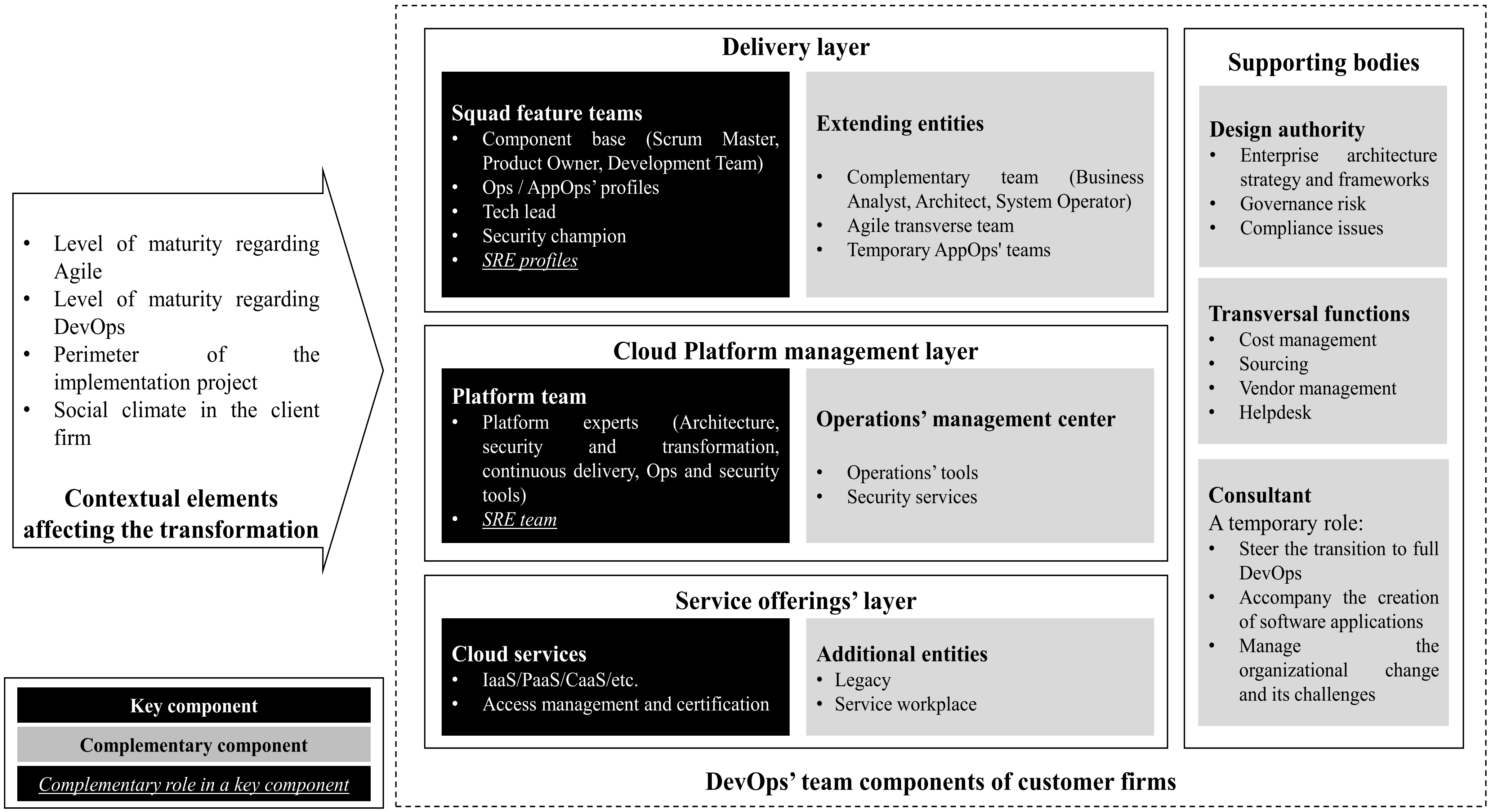
Zhu, L., Bass, L., & Champlin-Scharff, G. (2016). DevOps and its practices. IEEE Software, 33(3), 32-34.

8. Appendices

Appendix 1: Comparative summary of the research findings

Case	Firm properties	Project aim	Team structure	Change process
FinGov	*Sector*: Public financial administration *Size*: Around 100000 *Maturity*: No prior knowledge of DevOps and Agile	Experiment a new organization in the construction and the exploitation of its computer applications	• Minimum base: Product Owner, Scrum Master, Development team • Complementary team: Business Analyst, Architect, System Operator	Implementation of Agile then of DevOps Experimentation within 8 volunteer projects Resistance to change due to fear of DevOps' impact on the existence of job positions. Challenging implementation due to a tense social climate resulting from an important jobs' cut. Only four regional bodies transformed
TelCo	*Sector*: Virtual Telecom operator *Size*: 800 *Maturity*: Low familiarity. The Build works in Agile	Eliminate the break between the Build and the Run induced by the management of the Run by the historical host of the insurer.	Teams organized according to SAFe and including three operational layers: • A delivery layer composed of Feature teams responsible for software delivery (including a Product owner, a Scrum Master, a Tech lead, a Security champion, and AppOps' profiles) and of Transverse teams responsible for the Agile method • A cloud management layer composed of a platform team managing the infrastructure services and of an Operational security center responsible for responding to security incidents • A service offerings' layer including Cloud services, Service Workplace, and a Legacy entity Two structural bodies in parallel to operations: • A design authority ensuring technological consistency across all IT functions • Transversal functions (Costs, helpdesk, supplier management, etc.)	The change occurred gradually over two stages: • A first post-migration stage: Establishment of the central platform team, and of an AppOps' teams responsible for recuperating in Run the migrated applications on the cloud. • A second transition phase to full DevOps: Integration of AppOps into the feature teams in charge of application products
BigBank	*Sector*: Banking *Size*: >120000 *Maturity*: Highly familiar with DevOps. Used over 4 years	Define and set up SRE (Site Reliability Engineering) profiles i.e., development engineers at the service of operations	A DevOps' operating model composed of: • An upper layer associated with professional services, structured into sub-departments each one related to an element of BigBank activity and possessing its own feature team that includes SRE profiles. • A platform layer for cloud and operations management including a centralized SRE profile. • An offering layer including infrastructure models and a workplace service.	First, set up of SRE profiles in the feature teams to make them accountable of reliability issues. Then, implementation of central SRE profiles in the platform team to manage reliability of private and public clouds, handle incidents end-to-end, and define product operability standards.

Appendix 2: DevOps' team components and the determinants of structure variation


Level of maturity regarding Agile
Level of maturity regarding DevOps
Perimeter of the implementation project
Social climate in the client firm
Contextual elements affecting the transformation
Delivery layer
Squad feature teams
Component base (Scrum Master, Product Owner, Development Team)
Ops / AppOps' profiles
Tech lead
Security champion
SRE profiles
Extending entities
Complementary team (Business Analyst, Architect, System Operator)
Agile transverse team
Temporary AppOps' teams
Cloud Platform management layer
Platform team
Platform experts (Architecture, security and transformation, continuous delivery, Ops and security tools)
SRE team
Operations' management center
Operations' tools
Security services
Service offerings' layer
Cloud services
IaaS/PaaS/CaaS/etc.
Access management and certification
Additional entities
Legacy
Service workplace
Supporting bodies
Design authority
Enterprise architecture strategy and frameworks
Governance risk
Compliance issues
Transversal functions
Cost management
Sourcing
Vendor management
Helpdesk
Consultant
A temporary role:
Steer the transition to full DevOps
Accompany the creation of software applications
Manage the organizational change and its challenges
DevOps' team components of customer firms
Key component
Complementary component
Complementary role in a key component