## **ATLAS.ti** Report

### UnifiedDevOps OpenCoding ITE2 (ICA3)

### **Quotations grouped by Documents**

Report created by Isaque Alves on 29 Jul 2023

## 1 2021 DevOps Team Structures: Characterization and Implications

#### 29 Quotations:

# **● 1:1 p 5 in 2021 DevOps Team Structures: Characterization and Implications**

#### Codes:

- collaboration
- collaboration
- collaboration
- collaboration
- cross-functionality/skills
- cross-functionality/skills
- cross-functionality/skills
- cross-functionality/skills
- end-to-end product vision
- end-to-end product vision
- end-to-end product vision
- end-to-end product vision
- responsibility/ownership sharing
- responsibility/ownership sharing
- responsibility/ownership sharing
- responsibility/ownership sharing
- team self-organization & autonomy

#### Content:

Product team category emerges as a result of a set of codes that characterize these teams: collaboration, product ownership sharing, end-to-end product vision, crossfunctionality (sometimes used as synonym of multidisciplinary or poly�skilled teams), self-organization and autonomy

# **■** 1:2 p 5 in 2021 DevOps Team Structures: Characterization and Implications

#### Codes:

- leadership & management
- leadership & management
- leadership & management
- leadership & management
- role definition/attributions
- role definition/attributions
- role definition/attributions
- role definition/attributions
- small size teams (two pizza rule)

### Content:

Product teams are usually small (Amazon's two pizza rule) and are composed by high-qualiffed engineers and T-shape people. These teams promote skills over roles, leadership from management, and more frequently single management (referred to as product manager, product leader, technical leader, etc.) versus multivele management

# **⑤** 1:3 p 5 in 2021 DevOps Team Structures: Characterization and Implications

### Codes:

- collaboration
- collaboration
- collaboration
- collaboration
- cross-functionality/skills
- cross-functionality/skills
- cross-functionality/skills
- cross-functionality/skills

#### Content:

Product teams may involve those skills re lated to analysis, architecting and design, development, test ing, operation (system administration, monitoring), security, etc. or collaborate with other teams/departments that own some of these skills.

# **■ 1:4 p 5 in 2021 DevOps Team Structures: Characterization and Implications**

#### Codes:

- collaboration
- collaboration
- collaboration
- collaboration
- responsibility/ownership sharing
- responsibility/ownership sharing
- responsibility/ownership sharing
- responsibility/ownership sharing

#### Content:

the second one works closely with developers from the beginning of product development by establishing non-functional requirements (NFR shared responsibility), configuration ffles, deployment scripts, and other activities related to operations.

# 1:5 p 6 in 2021 DevOps Team Structures: Characterization and Implications

### Hyperlinks:

— continued by  $\rightarrow \textcircled{=} 1:6 \text{ p 6}$ , and lack of collaboration in 2021 DevOps Team Structures: Characterization and Implications

#### Codes:

- communication
- communication
- communication
- communication
- organizational silos/conflicts
- organizational silos/conflicts
- organizational silos/conflicts
- organizational silos/conflicts
- team self-organization & autonomy

### Content:

Some participating organizations highlighted that prodouct teams have external dependencies with other teams, mainly architecture, quality assurance, system administraotion, database administration, security, and ffrst-level operotions dependencies. These dependencies usually generate organizational barriers due to poor communication

# **■** 1:6 p 6 in 2021 DevOps Team Structures: Characterization and Implications

### Hyperlinks:

← continued by — ⑤ 1:5 p 6, Some participating organizations highlighted that prod � uct teams have external dependencies with oth... in 2021 DevOps Team Structures: Characterization and Implications

#### Codes:

- collaboration
- collaboration
- collaboration
- collaboration
- organizational silos/conflicts
- organizational silos/conflicts
- organizational silos/conflicts
- organizational silos/conflicts
- team self-organization & autonomy

### Content:

and lack of collaboration

# **■ 1:7 p 6 in 2021 DevOps Team Structures: Characterization and Implications**

### Codes:

- cultural silos/conflicts
- cultural silos/conflicts
- cultural silos/conflicts
- cultural silos/conflicts
- organizational silos/conflicts
- organizational silos/conflicts
- organizational silos/conflicts
- organizational silos/conflicts

#### Content

Some other organizations, although they addressed these organizational barriers, still show cultural barriers mainly between developers and operators (some times due to previous organizational silos that remain as vestigial cultural silos). Both organizational and cultural barriers are related to silos, which are instantiated as: op eration silos, system administration silos, security silos, quality assurance silos, architecture silos, and so on.

# **■** 1:8 p 6 in 2021 DevOps Team Structures: Characterization and Implications

#### Codes:

- collaboration
- collaboration
- collaboration
- collaboration
- organizational silos/conflicts
- organizational silos/conflicts
- organizational silos/conflicts
- organizational silos/conflicts

#### Content:

Emerging product teams resulting from the eventual inter�departmental collaboration between dev & ops and showing organizational silos

# **■** 1:9 p 6 in 2021 DevOps Team Structures: Characterization and Implications

### Hyperlinks:

— continued by → 

1:10 p 6, showing some cultural barriers. in 2021 DevOps Team Structures: Characterization and Implications

#### Codes:

- collaboration
- collaboration
- collaboration
- collaboration
- transfer of work between teams

#### Content:

Stable product teams resulting from the creation of teams in which developers and operators daily collaborate, but there exist still a transfer of work between them

# **■** 1:10 p 6 in 2021 DevOps Team Structures: Characterization and Implications

### Hyperlinks:

← continued by — 

1:9 p 6, Stable product teams resulting from the creation of teams in which developers and operators daily c... in 2021 DevOps Team Structures: Characterization and Implications

#### Codes:

- cultural silos/conflicts
- cultural silos/conflicts
- cultural silos/conflicts
- cultural silos/conflicts
- transfer of work between teams

#### Content:

showing some cultural barriers.

# **■ 1:11 p 6 in 2021 DevOps Team Structures: Characterization and Implications**

### Hyperlinks:

— continued by → ⑤ 1:12 p 6, shared product ownership, end-to-end product vision and high-levels of self-organization and autono… in 2021 DevOps Team Structures: Characterization and Implications

#### Codes:

- alignment of dev & ops goals
- cross-functionality/skills
- cross-functionality/skills
- cross-functionality/skills
- cross-functionality/skills
- cultural silos/conflicts
- cultural silos/conflicts
- cultural silos/conflicts
- organizational silos/conflicts
- organizational silos/conflicts
- organizational silos/conflicts

#### Content:

Consolidated product teams, which have dealt both organizational and cultural silos by aligning dev & ops goals with business goals and show cross-functional teams

## 1:12 p 6 in 2021 DevOps Team Structures: Characterization and **Implications**

### **Hyperlinks:**

← continued by — 

1:11 p 6, Consolidated product teams, which have dealt both organizational and cultural silos by aligning dev... in 2021 DevOps Team Structures: Characterization and Implications

#### Codes:

- cross-functionality/skills
- cultural silos/conflicts
- cultural silos/conflicts
- cultural silos/conflicts
- cultural silos/conflicts
- end-to-end product vision
- end-to-end product vision
- end-to-end product vision
- end-to-end product vision
- organizational silos/conflicts
- organizational silos/conflicts
- organizational silos/conflicts
- organizational silos/conflicts
- responsibility/ownership sharing
- responsibility/ownership sharing
- responsibility/ownership sharing
- responsibility/ownership sharing
- team self-organization & autonomy • team self-organization & autonomy
- team self-organization & autonomy
- team self-organization & autonomy

shared product ownership, end-to-end product vision and high-levels of selforganization and autonomy.

# **■** 1:13 p 6 in 2021 DevOps Team Structures: Characterization and Implications

### Hyperlinks:

- continued by → ⑤ 1:14 p 6, Platform servicing (e.g., CI/CD and releasing tools) and infrastructure (e.g., cloud infrastructu... in 2021 DevOps Team Structures: Characterization and Implications
- continued by → ⑤ 1:15 p 6, Evangelization and mentoring on DevOps practices for pro�moting culture values, such as communicat... in 2021 DevOps Team Structures: Characterization and Implications
- continued by → ⑤ 1:16 p 6, Rotary human resources, i.e., horizontal teams may facilitate and provide product teams with human... in 2021 DevOps Team Structures: Characterization and Implications

#### Codes:

- enabler (platform) team
- enabler (platform) team
- enabler (platform) team
- enabler (platform) team

### Content:

We also realized that these product teams are supported by horizontal (cross) teams, which may provide:

## **⑤** 1:14 p 6 in 2021 DevOps Team Structures: Characterization and Implications

### Hyperlinks:

← continued by — 

1:13 p 6, We also realized that these product teams are supported by horizontal (cross) teams, which may prov... in 2021 DevOps Team Structures: Characterization and Implications

#### Codes:

- enabler (platform) team
- enabler (platform) team
- enabler (platform) team
- enabler (platform) team
- platform servicing
- platform servicing
- platform servicing
- platform servicing
- values & best practices

#### Content:

- Platform servicing (e.g., CI/CD and releasing tools) and infrastructure (e.g., cloud infrastructure, virtualization or containerization, etc.) to implement best practices, such as continuous integration, continuous testing, continuous delivery and deployment, infrastructure as code, and continuous monitoring.

# **■** 1:15 p 6 in 2021 DevOps Team Structures: Characterization and Implications

### Hyperlinks:

← continued by — 

1:13 p 6, We also realized that these product teams are supported by horizontal (cross) teams, which may prov... in 2021 DevOps Team Structures: Characterization and Implications

#### Codes:

- enabler (platform) team
- enabler (platform) team
- enabler (platform) team
- enabler (platform) team
- evangelization and mentoring
- evangelization and mentoring
- evangelization and mentoring
- evangelization and mentoring
- knowledge sharing
- knowledge sharing
- knowledge sharing
- knowledge sharing
- values & best practices

### Content:

- Evangelization and mentoring on DevOps practices for pro�moting culture values, such as communication, transparency, and knowledge sharing.

# **■ 1:16 p 6 in 2021 DevOps Team Structures: Characterization and Implications**

### Hyperlinks:

← continued by — ⑤ 1:13 p 6, We also realized that these product teams are supported by horizontal (cross) teams, which may prov... in 2021 DevOps Team Structures: Characterization and Implications

#### Codes:

- cross-functionality/skills
- cross-functionality/skills
- enabler (platform) team
- enabler (platform) team
- enabler (platform) team
- enabler (platform) team
- rotary human resources
- rotary human resources
- rotary human resources
- rotary human resources
- values & best practices

#### Content:

Rotary human resources, i.e., horizontal teams may facilitate and provide product teams with human resources when these teams lack specific skills to undertake and accomplish their work and implement best practices.

# **■ 1:17 p 6 in 2021 DevOps Team Structures: Characterization and Implications**

#### **Hyperlinks:**

— continued by → ⑤ 1:18 p 6, Despite there are some differences among these teams, all of them refer to the same construct, i.e.... in 2021 DevOps Team Structures: Characterization and Implications

— continued by → ⑤ 1:19 p6, and/or mentoring among others, as a service in 2021 DevOps Team Structures: Characterization and Implications

#### Codes:

- enabler (platform) team
- enabler (platform) team
- enabler (platform) team
- enabler (platform) team

#### Content:

While using these codes, we also realized that there were different kinds of horizontal teams such as DevOps Center of Excellence (DevOps CoE), DevOps chapter and Platform team.

# **■** 1:18 p 6 in 2021 DevOps Team Structures: Characterization and Implications

### Hyperlinks:

← continued by — 

1:17 p 6, While using these codes, we also realized that there were different kinds of horizontal teams such... in 2021 DevOps Team Structures: Characterization and Implications

#### Codes:

- enabler (platform) team
- enabler (platform) team
- enabler (platform) team
- enabler (platform) team
- platform servicing
- platform servicing
- platform servicing
- platform servicing

#### Content:

Despite there are some differences among these teams, all of them refer to the same construct, i.e., teams that provide platform, infrastructure, IT operation,

# **■** 1:19 p 6 in 2021 DevOps Team Structures: Characterization and Implications

### Hyperlinks:

← continued by — 

1:17 p 6, While using these codes, we also realized that there were different kinds of horizontal teams such... in 2021 DevOps Team Structures: Characterization and Implications

#### Codes

- enabler (platform) team
- enabler (platform) team
- enabler (platform) team
- enabler (platform) team
- evangelization and mentoring
- evangelization and mentoring
- evangelization and mentoring
- evangelization and mentoring

### Content:

and/or mentoring among others, as a service

# **■** 1:20 p 6 in 2021 DevOps Team Structures: Characterization and Implications

#### Codes:

- enabler (platform) team
- enabler (platform) team
- enabler (platform) team
- enabler (platform) team
- evangelization and mentoring
- evangelization and mentoring
- evangelization and mentoring
- evangelization and mentoring
- platform servicing
- platform servicing
- platform servicing
- platform servicing
- rotary human resources
- rotary human resources
- team self-organization & autonomy
- team self-organization & autonomy

#### Content:

This gives autonomy to product teams. We wrote some memos to clarify this meaning as follows: MEMO: Horizontal (cross) DevOps teams, either DevOps CoE, DevOps chapter, or Platform team, aim to provide a DevOps platform, IT operation, or mentoring, for autonomous product teams. They own DevOps skills & culture, platform, tools, and infrastructure to provide product teams with (i) servicing of CI/CD platform and environments for dev, test, or even pre-production, (ii) mentoring and evangelizing, and sometimes (iii) engineers who get involved in product teams with exclusive dedication but limited in time until these teams are capable of the "you build it, you run it".

# **■ 1:21 p 6 in 2021 DevOps Team Structures: Characterization and Implications**

#### Codes:

- cross-functionality/skills
- cross-functionality/skills
- enabler (platform) team
- enabler (platform) team
- enabler (platform) team
- enabler (platform) team
- rotary human resources
- rotary human resources
- rotary human resources
- rotary human resources

### Content:

Figure 1 shows a cross team composed of high qualiffed engineers in DevOps culture, speciffcally 5 senior developers, 10 testers and quality assurance engineers, and 10 IT operators (second-level operations), who get in volved in product teams when necessary. These engineers are involved in product teams with exclusive dedication but limited in time, until product teams are capable of doing all their responsibilities, from planning, analysis, development, testing, deployment, to operation. This means that these horizontal teams are composed of engineers that move through the product teams according to their needs. The reason that these engineers are not part of the product teams is that these organizations (like ID2) do not have human resources enough to involve the necessary engineers in all the product teams.

# **■** 1:22 pp 6 – 7 in 2021 DevOps Team Structures: Characterization and Implications

#### Codes:

- enabler (platform) team
- enabler (platform) team
- enabler (platform) team
- enabler (platform) team
- platform servicing
- platform servicing
- platform servicing
- platform servicing
- team self-organization & autonomy
- team self-organization & autonomy
- team self-organization & autonomy

#### Content:

Figure 2 shows an example in which the operations department assumes the DevOps culture, provides to developers (Scrum teams) with platforms and infrastructure, and enables scrum teams to be autonomous. This means, the operations department assumes the functions of a DevOps platform team. This example differs from the previous one Authorized licensed use limited to: Univ Politecnica de Madrid. Downloaded on August 16,2021 at 11:00:44 UTC from IEEE Xplore. Restrictions apply.

0098-5589 (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See

http://www.ieee.org/publications\_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSE.2021.3102982, IEEE Transactions on Software Engineering JOURNAL OF LATEX CLASS FILES, VOL. 14, NO. 8, AUGUST 2015 7 Fig. 1. Organizational team structure by ID2 in the fact that there is no immersion of engineers from the horizontal team to the product teams.

# **■ 1:23 p 7 in 2021 DevOps Team Structures: Characterization and Implications**

#### Codes:

- collaboration
- collaboration
- collaboration
- collaboration
- enabler (platform) team
- enabler (platform) team
- enabler (platform) team
- enabler (platform) team
- evangelization and mentoring
- evangelization and mentoringevangelization and mentoring
- evangenzation and mentoring
- evangelization and mentoring
- platform servicing
- platform servicing
- platform servicingplatform servicing

#### Content:

Figure 3 shows another approach in which a horizontal team (i) develops, in collaboration with the rest of the departments, a DevOps platform for internal use, and (ii) evangelizes DevOps practices. This example differs from the previous ones in the fact that the horizontal team behaves as a product team (the product is the DevOps platform) while it provides services to both product teams and classical operations (either cloud or on-premise).

# **⑤** 1:24 p 7 in 2021 DevOps Team Structures: Characterization and Implications

### Codes:

- enabler (platform) team
- enabler (platform) team
- enabler (platform) team
- enabler (platform) team
- team self-organization & autonomy
- team self-organization & autonomy
- team self-organization & autonomy

### Content:

Finally, the following excerpts show more evidence of the existence of horizontal DevOps teams in the participat ing organizations and its importance to make product teams autonomous.

# **■ 1:25 p 7 in 2021 DevOps Team Structures: Characterization and Implications**

#### Codes:

- cross-functionality/skills
- cross-functionality/skills
- cross-functionality/skills
- responsibility/ownership sharing
- responsibility/ownership sharing
- responsibility/ownership sharing
- responsibility/ownership sharing
- small size teams (two pizza rule)

#### Content:

3.2.1 Product ownership sharing We observed that there exists a relationship between how product teams share the product ownership and how these teams are structured. For example, ID29 shows a high level of sharing of the product ownership within product teams, which are cohesive, small (less than 12 people), and multidisciplinary

## **■ 1:26 p 8 in 2021 DevOps Team Structures: Characterization and Implications**

#### Codes:

- end-to-end product vision
- leadership & management
- leadership & management
- leadership & management
- leadership & management
- responsibility/ownership sharing
- responsibility/ownership sharing
- responsibility/ownership sharing
- responsibility/ownership sharing

#### Content:

3.2.2 Leadership from management We realized that shared ownership is highly related to lead ership from management, which is an interesting variable to be examined due to its impact on organizational team structures. Hence, non-shared leadership usually leads to non-shared ownership because, if there are multiple man agers within the same team (typically one development manager and one operation manager), then it is difffcult for all members to feel the product as a whole. This means that each member tends to take only a part of the responsibility (developers give priority to develop new features whereas operators give priority to service stability).

# **■ 1:27 p 8 in 2021 DevOps Team Structures: Characterization and Implications**

#### Codes:

- cultural silos/conflicts
- cultural silos/conflicts
- cultural silos/conflicts
- cultural silos/conflicts
- organizational silos/conflicts
- organizational silos/conflicts
- organizational silos/conflicts
- organizational silos/conflicts

#### Content:

3.2.3 Organizational silos & Cultural silos We also found that the existence of strong hierarchical organization charts and departmental structure impact the structure of teams because both organizational and cultural silos undermined the adoption of DevOps practices and culture. The structure of some organizations like ID01 leads to the creation of silos and ffnd themselves with serious problems to adopt DevOps. In other cases like ID27 the or ganizational silos were broken, but the cultural ones remain (at least for a while) hindering the DevOps adoption. Many organizations like ID29 have managed to transform their structure, eliminated all the silos and achieved a complete adoption of DevOps. Recently founded organizations like ID09 usually do not face silo problems because they were born with a structure that favors DevOps. By observing the big picture, we established two levels on the organizational silo variable: yes, no; and three levels on the cultural silo variable: yes, no, vestigial (previous silos remain as vestigial cultural silos).

# **■ 1:28 p 8 in 2021 DevOps Team Structures: Characterization and Implications**

#### Codes:

- collaboration
- collaboration
- collaboration
- collaboration
- role definition/attributions
- role definition/attributions
- role definition/attributions
- role definition/attributions
- transfer of work between teams

#### Content:

3.2.4 Collaboration We also noticed that collaboration frequency is highly related to the team structures and is a critical variable for DevOps adoption. Indeed, collaboration is one of the key values of DevOps culture. Hence, the members of product teams may work together regularly on a daily basis to undertake all the product life-cycle, as it happens in orgalizations like ID23. This implies a daily collaboration believen team members and usually a daily meeting, without detriment of other less frequent meetings with other related teams. However, the members of product teams in other organizations have more differentiated roles (dev versus ops) so that they work together but in different tasks. This means that there is not a real collaboration, but a transfer of responsibilities, as it happens in organizations like ID05. In these cases, the collaboration frequency is on a weekly basis or even more. In this way, we established three levels on the collaboration frequency variable: daily, frequent and eventual

# **■** 1:29 pp 8 – 9 in 2021 DevOps Team Structures: Characterization and Implications

#### Codes:

- automated application life-cycle management
- delivery performance
- end-to-end product vision
- end-to-end product vision
- end-to-end product vision
- team self-organization & autonomy

#### Content:

3.2.5 Autonomy Last but not least, we found that autonomy might reveal the organizational team structure of a company. We under stand that a DevOps product team is entirely autonomous when it does not have external dependencies to fulffll its responsibilities, this implies having an end-to-end vision Authorized licensed use limited to: Univ Politecnica de Madrid. Downloaded on August 16,2021 at 11:00:44 UTC from IEEE Xplore. Restrictions apply.

0098-5589 (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See

http://www.ieee.org/publications\_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSE.2021.3102982, IEEE Transactions on Software Engineering JOURNAL OF LATEX CLASS FILES, VOL. 14, NO. 8, AUGUST 2015 9 and taking complete charge of a product from its conception and implementation to its deployment and monitoring. This is hard to achieve and we only found a few organizations like ID03 whose product teams implement the practice con tinuous deployment and continuous feedback, and thus, being completely autonomous. The most common practice is that product teams implement continuous delivery so that they can deploy in a pre-production environment, but they need external approval to go into production. These approvals may come directly from business or from technical areas such as quality or security. This was very usual in most organizations like ID08, even if their DevOps maturity was high. However, in some organizations, product teams still have many dependencies and

they do not manage continuous delivery, much less continuous deployment.

■ 2 2021 The organization of software teams in the quest for continuous delivery: A grounded theory approach

42 Quotations:

# 2:1 p 6 in 2021 The organization of software teams in the quest for continuous delivery: A grounded theory approach

#### Codes:

- alignment of dev & ops goals
- communication
- communication
- communication
- communication
- organizational silos/conflicts
- organizational silos/conflicts
- organizational silos/conflicts
- organizational silos/conflicts

#### Content:

With siloed departments, developers and the infrastructure staff are segregated from each other, with little direct communication among them. Frictions occurs among silos, since developers want to deliver as much as possible, whereas operations target stability and block developers.

## 2:2 p 6 in 2021 The organization of software teams in the quest for continuous delivery: A grounded theory approach

#### Codes:

- collaboration
- collaboration
- collaboration
- role definition/attributions
- role definition/attributions
- role definition/attributions
- role definition/attributions
- transfer of work between teams

#### Content:

→ Developers and operators have well-defined and differentiated roles; as stated by #120: "the wall was very clear: after committing, our work [as developers] was done". Therefore, there are no conflicts concerning attributions. Well-defined roles and pipelines can decrease the need for inter-departmental direct collaboration (#110).

# 2:3 p 6 in 2021 The organization of software teams in the quest for continuous delivery: A grounded theory approach

#### Codes:

- alignment of dev & ops goals
- collaboration
- organizational silos/conflicts
- organizational silos/conflicts

#### Content:

→ Each department is guided by its own interests, looking for local optimization rather than global optimization, an old and problem atic pattern [37]. Participant #126 told us "there is a big war there. . . the security, governance, and audit groups must still be convinced that the tool [Docker/Kubernetes] is good".

# 2:4 p 6 in 2021 The organization of software teams in the quest for continuous delivery: A grounded theory approach

#### Codes:

- knowledge sharing
- knowledge sharing
- organizational silos/conflicts
- organizational silos/conflicts
- organizational silos/conflicts

### Content:

→ Developers have minimal awareness of what happens in production (#I26). So monitoring and handling incidents are mostly done by the infrastructure team (#I5).

# 2:5 p 6 in 2021 The organization of software teams in the quest for continuous delivery: A grounded theory approach

### Codes:

- alignment of dev & ops goals
- collaboration
- collaboration
- organizational silos/conflicts
- organizational silos/conflicts
- responsibility/ownership sharing
- responsibility/ownership sharing
- responsibility/ownership sharing

#### Content:

Developers often neglect non-functional requirements (NFRs), especially security (#I5). In #I30, conflicts among developers and the security group arise from disagreement on technical decisions.

In other cases, developers have little contact with the security group (#I26).

# **2:6 p 6 in 2021 The organization of software teams in the quest for continuous delivery: A grounded theory approach**

#### Codes:

- automated application life-cycle management
- automated application life-cycle management
- automated application life-cycle management
- communication
- communication
- communication
- communication

#### Content:

Limited DevOps initiatives, centered on adopting tools, do not improve communication

# © 2:7 p 6 in 2021 The organization of software teams in the quest for continuous delivery: A grounded theory approach

#### Codes:

- automated application life-cycle management
- automated application life-cycle management
- automated application life-cycle management
- collaboration
- collaboration
- collaboration
- collaboration
- knowledge sharing

### Content:

collaboration among teams (#I30) or spread awareness about automated tests (#I5, #I15).

# 2:8 p 6 in 2021 The organization of software teams in the quest for continuous delivery: A grounded theory approach

#### Codes:

- automated application life-cycle management
- automated application life-cycle management
- automated application life-cycle management
- organizational silos/conflicts
- organizational silos/conflicts
- organizational silos/conflicts
- organizational silos/conflicts

### Content:

In #I30, a "De�vOps team" maintaining the deployment pipeline behaves as another silo, sometimes bottlenecking the delivery [38].

# 2:9 p 7 in 2021 The organization of software teams in the quest for continuous delivery: A grounded theory approach

#### Codes:

- delivery performance
- delivery performance
- delivery performance
- delivery performance
- organizational silos/conflicts
- team self-organization & autonomy
- team self-organization & autonomy
- team self-organization & autonomy

#### Content:

Organizations are less likely to achieve high delivery perfor mance as developers need bureaucratic approval to deploy applications and evolve the database schema (#I5, #I30). Table 3 shows that only two of 13 siloed organizations presented high delivery performance, and these two were already transitioning to other structures. However, we observed cases in which low delivery performance was not a probelem, such as short-lived research experiments (#I13) and releases of new phases of a game not requiring code changes (#I10). Network isolation policies may also hinder frequent deployment (#B1, #I7).

## 2:10 p 7 in 2021 The organization of software teams in the quest for continuous delivery: A grounded theory approach

#### Codes:

- automated application life-cycle management
- organizational silos/conflicts
- organizational silos/conflicts

#### Content:

→ We observed a lack of proper test automation in many orga�nizations (#15, #115, #123, #126). In #126, developers automate only unit tests. Organization #115 was leaving test automation only for QA people, which is not suitable for TDD or unit tests. Although siloed organizations are not the only ones that lack test automation (#13, #132, #135), in this structure developers can even ignore its value (#15, #123, #137). We notice that some of the observed scenarios were more challenging for test automation, such as games.

# 2:11 p 7 in 2021 The organization of software teams in the quest for continuous delivery: A grounded theory approach

#### Codes:

- collaboration
- collaboration
- collaboration
- collaboration
- organizational silos/conflicts
- organizational silos/conflicts

#### Content:

The classical DevOps structure focuses on collaboration among developers and the infrastructure team. It does not eliminate all conflicts, but promotes a better environment to deal with them (#I34). We named this structure "Classical DevOps" because we understand that a collaborative culture is the core DevOps concern [23,26,39]. We classified ten organizations into this structure. We also observed three organizations transitioning to this structure and three transitioning out of this structure.

## 2:12 p 8 in 2021 The organization of software teams in the quest for continuous delivery: A grounded theory approach

#### Codes:

- collaboration
- collaboration
- collaboration
- collaboration
- · responsibility/ownership sharing
- responsibility/ownership sharing
- responsibility/ownership sharing
- role definition/attributions
- role definition/attributions

#### Content:

→ We observed that, in classical DevOps settings, many practices foster a culture of collaboration. We saw the sharing of database management: infrastructure staff creates and fine tunes the database, whereas developers write queries and manage the database schema (#I17).

# 2:13 p 8 in 2021 The organization of software teams in the quest for continuous delivery: A grounded theory approach

#### Codes:

- · alignment of dev & ops goals
- communication
- communication
- communication
- communication
- responsibility/ownership sharing
- responsibility/ownership sharing

#### Content:

We heard about open communication among developers and the infrastructure team (#I2, #I6, #I17, #I22, #I31, #I36). Participant #I2 highlighted that: "Development and infrastructure teams participate in the same chat; it even looks like everyone is part of the same team".

Developers also support the product in its initial production (#I31).

## 2:14 p 8 in 2021 The organization of software teams in the quest for continuous delivery: A grounded theory approach

### Codes:

- collaboration
- collaboration
- collaboration
- collaboration
- cross-functionality/skills
- role definition/attributions
- role definition/attributions
- role definition/attributions

### Content:

 $\rightarrow$  Roles remain well-defined, and despite the collaboration on some activities, there are usually no conflicts over who is responsible for each task.

# 2:15 p 8 in 2021 The organization of software teams in the quest for continuous delivery: A grounded theory approach

#### Codes:

- cross-functionality/skills
- cross-functionality/skills
- organizational silos/conflicts
- organizational silos/conflicts
- role definition/attributions
- transfer of work between teams

#### Content:

→ Developers feel relieved when they can rely on the infrastruc ture team (#I17). Participant #I31 claimed that his previous job in a cross-functional team had a much more stressful environment than his current position in a development team in a classical DevOps environment. On the other hand, stress can persist at high levels for the infrastructure team (#I34), especially "if the application is ill-designed and has low performance" (#I36).

# 2:16 p 8 in 2021 The organization of software teams in the quest for continuous delivery: A grounded theory approach

#### Codes:

- alignment of dev & ops goals
- role definition/attributions
- role definition/attributions

#### Content:

→ In this structure, the project's success depends on the alignment of different departments, which is not trivial to achieve. In #B3, dif�ferent teams understood the organization's goals and the consequences of not solving problems, like wrongly computing amounts in the order of millions of dollars. Moreover, #I7 described that alignment emerges when employees focus on problem-solving rather than role attributions.

## 2:17 p 8 in 2021 The organization of software teams in the quest for continuous delivery: A grounded theory approach

#### Codes:

- · responsibility/ownership sharing
- responsibility/ownership sharing
- responsibility/ownership sharing
- responsibility/ownership sharing

### Content:

→ Development and infrastructure teams share NFR responsi�bilities (#17). For example, in #I2, both were very concerned with low latency, a primary requirement for their application.

# 2:18 p 8 in 2021 The organization of software teams in the quest for continuous delivery: A grounded theory approach

### Codes:

- collaboration
- collaboration
- collaboration
- collaboration
- responsibility/ownership sharing
- responsibility/ownership sharing
- responsibility/ownership sharing
- responsibility/ownership sharing
- role definition/attributions

### Content:

 $\rightarrow$  Usually, the infrastructure staff is the front line of tracking monitoring and incident handling (#I2, #I11, #I29, #I31, #I36).

However, if needed, developers are summoned and collaborate (#117, #134). In #134, monitoring alerts are directed to the infrastructure team but copied to developers. However, in some cases developers never work after-hours (#12, #122).

# 2:19 p 8 in 2021 The organization of software teams in the quest for continuous delivery: A grounded theory approach

#### Codes:

- collaboration
- collaboration
- collaboration
- collaboration
- organizational silos/conflicts
- organizational silos/conflicts

#### Content:

Humble expects a culture of collaboration among developers and the infrastructure staff to prescind from a "DevOps team" [38].

We understand this criticism applies to DevOps teams with dedicated members, such as we saw in #I30, since they behave as new silos.

## 2:20 p 8 in 2021 The organization of software teams in the quest for continuous delivery: A grounded theory approach

#### Codes:

- enabler (platform) team
- enabler (platform) team
- knowledge sharing
- knowledge sharing
- knowledge sharing
- knowledge sharing
- leadership & management
- leadership & management
- leadership & management
- leadership & management

#### Content

However, we found in #I36 a well-running DevOps team working as a committee for strategic decisions — a forum for the leadership of different departments. We also found DevOps groups working as guilds (#I4, #I8), supporting knowledge exchange among different departments [40]

# 2:21 p 8 in 2021 The organization of software teams in the quest for continuous delivery: A grounded theory approach

#### Codes:

- automated application life-cycle management
- collaboration
- collaboration
- collaboration
- collaboration
- delivery performance
- delivery performance
- delivery performance
- delivery performance

#### Content:

→ Collaboration and delivery automation, critical values of the DevOps movement, are not enough to achieve high delivery per�formance.

## 2:22 p 8 in 2021 The organization of software teams in the quest for continuous delivery: A grounded theory approach

#### Codes:

- automated application life-cycle management
- delivery performance
- delivery performance
- delivery performance
- delivery performance

#### Content:

Of 10 classical DevOps organizations not transitioning from or to other structures, only three presented high delivery performance (Table 3). One possible reason is the lack of proper test automation (#I22, #I36) [41]. Another limitation for delivery performance is the adoption of release windows (#I11, #I31, #I14, #I36), which seek to mitigate deployment risk by restricting software delivery to periodic time slots. Release windows are adopted by considering either the massive number of users (#I31) or the system's financial criticality (#I36). Release windows may also result from fragile architectures (#I37) or the monolith architectural style (#I11) since any deployment has an increased risk of affecting the whole system

# 2:23 p 8 in 2021 The organization of software teams in the quest for continuous delivery: A grounded theory approach

#### Codes:

- collaboration
- collaboration
- collaboration
- collaboration
- responsibility/ownership sharing
- responsibility/ownership sharing
- responsibility/ownership sharing
- role definition/attributions

#### Content:

Infra as development collaborator. The infrastructure staff con tributes to the application code to optimize the system's performance, reliability, stability, and availability. Although this aptitude requires advanced coding skills from infrastructure professionals, it is a suitable strategy for maintaining large-scale systems, like the ones owned by #I31.

# 2:24 p 8 in 2021 The organization of software teams in the quest for continuous delivery: A grounded theory approach

#### Codes:

- cross-functionality/skills
- cross-functionality/skills
- cross-functionality/skills
- cross-functionality/skills
- responsibility/ownership sharing
- responsibility/ownership sharing
- team self-organization & autonomy

### Content:

In our context, a cross-functional team takes responsibility for both software development and infrastructure management. This structure aligns with the Amazon motto "You built it, you run it" [42] and with the "autonomous squads" at Spotify [40]. This gives more freedom to the team, along with a great deal of responsibility. As interviewee #I1 described: "it is like each team is a different mini-company, having the freedom to manage its own budget and infrastructure".

# 2:25 p 8 in 2021 The organization of software teams in the quest for continuous delivery: A grounded theory approach

#### Codes:

- alignment of dev & ops goals
- communication
- communication
- communication
- communication
- cross-functionality/skills
- cross-functionality/skills
- cross-functionality/skills
- cross-functionality/skills

#### Content:

→ Independence among teams may lead to misalignment. Lack of communication and standardization among cross-functional teams within a single organization may lead to duplicated efforts (#I28).

However, this is not always a problem (#I1).

## 2:26 p 8 in 2021 The organization of software teams in the quest for continuous delivery: A grounded theory approach

#### Codes:

- automated application life-cycle management
- automated application life-cycle management
- cross-functionality/skills
- cross-functionality/skills
- cross-functionality/skills
- cross-functionality/skills
- knowledge sharing

### Content:

→ It is hard to ensure a team has all the necessary skills. For instance, we interviewed two cross-functional teams with no infrastruc ture expertise (#I3, #I32). Participant #I27 recognizes that "there is a lack of knowledge" on infrastructure, deployment automation, and monitoring.

# 2:27 p 8 in 2021 The organization of software teams in the quest for continuous delivery: A grounded theory approach

#### Codes:

- cross-functionality/skills
- cross-functionality/skills
- knowledge sharing
- role definition/attributions
- role definition/attributions

#### Content:

A possible reason for such adversity is that, as #I29 taught us, it is hard to hire infrastructure specialists and senior developers.

→ We expected cross-functional teams to provide too much idle time for specialists, as opposed to centralized pools of specialization.

However, we find no evidence of idleness for specialists. From #I16, we heard quite the opposite: the infrastructure specialists were too busy to be shared with other teams. Having the infrastructure specialists code features in their spare time avoids such idleness (#I35).

# 2:28 p 8 in 2021 The organization of software teams in the quest for continuous delivery: A grounded theory approach

#### Codes:

- cross-functionality/skills
- cross-functionality/skills
- cross-functionality/skills
- cross-functionality/skills
- small size teams (two pizza rule)
- small size teams (two pizza rule)

#### Content:

→ Most of the cross-functional teams we interviewed were in small organizations (Table 4), likely because there is no sense in creating multiple teams in too small organizations.

Supplementary properties

# 2:29 p 8 in 2021 The organization of software teams in the quest for continuous delivery: A grounded theory approach

#### Codes:

- cross-functionality/skills
- cross-functionality/skills
- cross-functionality/skills
- cross-functionality/skills

### Content:

Dedicated infra professionals. The team has specialized people dedicated to infrastructure tasks. In #I1, one employee specializes in physical infrastructure, and another is "the DevOps", taking care of the deployment pipeline and monitoring. In this circumstance, the infrastructure specialists become the front-line for tackling incidents and monitoring (#I28, #I35).

# **②** 2:30 pp 8 − 9 in 2021 The organization of software teams in the quest for continuous delivery: A grounded theory approach

#### Codes:

- responsibility/ownership sharing
- responsibility/ownership sharing
- role definition/attributions
- role definition/attributions
- role definition/attributions
- role definition/attributions

#### Content:

Developers with infra background. The team has developers knowledgeable in infrastructure management; these professionals are also called full-stack engineers or even DevOps engineers (#I25).

Participant #125 is a full-stack engineer and claimed to "know all the Information and Software Technology 139 (2021) 106672 9 L. Leite et al. Table 4 Organizational structures and organization size observed in our interviews. Organizational structure Organization size Number of interviews Siloed departments < 200 3 Siloed departments > 1000 4 Classical DevOps < 200 2 Classical DevOps > 200 and < 1000 4 Classical DevOps > 1000 4 Cross-functional < 200 5 Crossfunctional > 200 and < 1000 2 Platform team > 200 and < 1000 2 Platform team > 1000 2 Siloed departments > 200 and < 1000 2 to Classical DevOps Siloed departments > 1000 1 to Classical DevOps Siloed departments > 200 and < 1000 1 to Cross-functional Siloed departments > 1000 2 to Platform team Classical DevOps < 200 1 to Cross-functional Classical DevOps > 200 and < 1000 1 to Platform team Cross-functional > 1000 1 to Platform team involved technologies: front-end, backend, and infrastructure; so I am the person able to link all of them and to firefight when needed". Participant #I29, a consultant, is skeptical regarding full-stack engineers and stated that "these people are not up to the task". He complained that developers are usually unaware of how to fine tune the application, such as configuring database connections.

## 2:31 p 9 in 2021 The organization of software teams in the quest for continuous delivery: A grounded theory approach

### Codes:

- cross-functionality/skills
- cross-functionality/skills
- cross-functionality/skills
- cross-functionality/skills
- responsibility/ownership sharing

### Content:

No infra background. Product teams manage the infrastructure without the corresponding expertise. We saw this pattern in two places.

One was a very small company and had just released their application, having only a few users (#I32) and being uncertain about hiring specialized people soon. Interviewee #I3 understands that operations work (e.g., spotting errors during firmware updates in IoT devices and starting Amazon VMs for new clients) is too menial for software engitherers, taking too much of their expensive time. So the organization was planning the creation of an operations sector composed of a cheaper workforce. Interviewee #I19 argued that such an arrangement could not sustain growth in his company in the past.

# 2:32 p 9 in 2021 The organization of software teams in the quest for continuous delivery: A grounded theory approach

#### Codes:

- automated application life-cycle management
- automated application life-cycle management
- enabler (platform) team
- enabler (platform) team
- enabler (platform) team
- enabler (platform) team
- platform servicing
- platform servicing

#### Content:

Platform teams are infrastructure teams that provide highly auto@mated infrastructure services that can be self-serviced by developers for application deployment. The infrastructure team is no longer a "support team"; it behaves like a product team, with the "platform" as its product and developers as internal customers.

# 2:33 p 9 in 2021 The organization of software teams in the quest for continuous delivery: A grounded theory approach

#### Codes:

- enabler (platform) team
- enabler (platform) team
- platform servicing
- platform servicing
- · responsibility/ownership sharing
- responsibility/ownership sharing
- responsibility/ownership sharing
- responsibility/ownership sharing

### Content:

Product teams are fully accountable for the non-functional requirements of their services. They become the first ones called when there is an incident, which is escalated to the infrastructure people only if the problem relates to an infrastructure service (#18, #19, #112, #133).

→ Although the product team becomes fully responsible for NFRs of its services, it is not a significant burden that developers try to refuse (#I33). The platform itself handles many NFR concerns, such as load balancing, auto-scaling, throttling, and high-speed communica tions between data-centers (#I4, #I8, #I16, #I33). As participant #I33 told us, "you do not need to worry about how things work, they just work".

Moreover, we observed infrastructure people willingly supporting de velopers for the sake of services availability, performance, and security (#I9, #I14).

# 2:34 p 9 in 2021 The organization of software teams in the quest for continuous delivery: A grounded theory approach

#### Codes:

- communication
- communication
- communication
- communication
- enabler (platform) team
- enabler (platform) team
- platform servicing
- platform servicing
- responsibility/ownership sharing

#### Content:

19, #I14).

→ Product teams become decoupled from the members of the platform team. Usually, the communication among these teams hap pens when developers and infrastructure people gather to solve incidents (#I8, #I9); when infrastructure people provide consulting for developers to master non-functional concerns (#I9); or when develop pers demand new capabilities from the platform (#I8, #I12).

# 2:35 p 9 in 2021 The organization of software teams in the quest for continuous delivery: A grounded theory approach

#### Codes:

- collaboration
- collaboration
- collaboration
- collaboration
- enabler (platform) team
- enabler (platform) team
- platform servicing

#### Content:

In this way, the decoupling between the platform and product teams does not imply the absence of collaboration among these groups.

# 2:36 p 9 in 2021 The organization of software teams in the quest for continuous delivery: A grounded theory approach

#### Codes:

- automated application life-cycle management
- automated application life-cycle management
- enabler (platform) team
- platform servicing
- platform servicing

#### Content:

The infrastructure team is no longer requested for opera tional tasks. The operational tasks are automated by the platform.

Therefore, one cannot merely call platform-team members "operators", since they also engineer the infrastructure solution.

# 2:37 p 9 in 2021 The organization of software teams in the quest for continuous delivery: A grounded theory approach

#### Codes:

- cross-functionality/skills
- cross-functionality/skills
- enabler (platform) team
- enabler (platform) team
- knowledge sharing
- role definition/attributions
- role definition/attributions

### Content:

→ The platform avoids the need for product teams to have in� frastructure specialists. Participant #I33 expressed wanting to better understand what happens "under the hood" of the platform, which indicates how well the platform relieves the team from mastering infrastructure concerns. On the other hand, since developers are re� sponsible for the deployment, they must have some basic knowledge about the infrastructure and the platform itself.

## 2:38 p 9 in 2021 The organization of software teams in the quest for continuous delivery: A grounded theory approach

#### Codes:

- cross-functionality/skills
- cross-functionality/skills
- cross-functionality/skills
- enabler (platform) team
- enabler (platform) team
- enabler (platform) team
- enabler (platform) team
- platform servicing

### Content:

→ The platform may not be enough to deal with particular requirements. Participant #I16 stated that "if a lot of people do similar functionality, over time usually it gets integrated to the platform. . . but each team will have something very specialized. . . " to explain the presence of infrastructure staff within the team, even with the usage of a platform, considering the massive number of virtual machines to be managed.

# 2:39 p 9 in 2021 The organization of software teams in the quest for continuous delivery: A grounded theory approach

#### Codes:

- cross-functionality/skills
- cross-functionality/skills
- cross-functionality/skills
- enabler (platform) team
- platform servicing
- platform servicing
- platform servicing
- platform servicing

#### Content:

→ If the organization develops a new platform to deal with its specificities, it will require development skills from the infrastruc ture team. Nevertheless, even without developing a new platform, the infrastructure team must have a "dev mindset" to produce scripts and use infrastructure-as-code [43] to automate the delivery path (#114).

One strategy we observed to meet this need was to hire previous developers for the infrastructure team (#I14).

# 2:40 p 9 in 2021 The organization of software teams in the quest for continuous delivery: A grounded theory approach

#### Codes:

- delivery performance
- enabler (platform) team
- enabler (platform) team
- enabler (platform) team
- enabler (platform) team
- responsibility/ownership sharing
- responsibility/ownership sharing
- team self-organization & autonomy
- team self-organization & autonomy
- team self-organization & autonomy

#### Content:

→ All four organizations that have fully embraced the platform team structure are high performers, while no other structure pro�vided such a level of success (Table 3). An explanation for such a relation is that this structure decouples the infrastructure and product teams, which prevents the infrastructure team from bottlenecking the delivery path. As stated by #I20: "Now developers have autonomy for going from zero to production without having to wait for anyone". This structure also contributes to service reliability by letting product teams handle non-functional requirements and incidents

# 2:41 p 10 in 2021 The organization of software teams in the quest for continuous delivery: A grounded theory approach

#### Codes:

- automated application life-cycle management
- enabler (platform) team
- enabler (platform) team
- knowledge sharing
- knowledge sharing
- knowledge sharing

#### Content:

Enabler team. An enabler team provides consulting and tools for product teams but does not own any service. Consulting can be on per�formance (#I18) or security (#I9, #I16, #I31), for example. Tools pro�vided by enabler teams include the deployment pipeline (#I4, #I30), high-availability mechanisms (#I11), monitoring tools (#I12), and se�curity tools (#I17).

# 2:42 p 10 in 2021 The organization of software teams in the quest for continuous delivery: A grounded theory approach

#### Codes:

- automated application life-cycle management
- automated application life-cycle management
- automated application life-cycle management
- collaboration
- collaboration
- collaboration
- collaboration
- cross-functionality/skills
- cross-functionality/skills
- cross-functionality/skills
- organizational silos/conflicts
- organizational silos/conflicts
- platform servicing

#### Content:

With a platform. The organization possesses a platform that can provide deployment automation, but not following the patterns of human interaction and collaboration described by the core properties of platform teams. Participant #I25 developed an "autonomous IaaC for integration and deployment with Google Cloud", which provides a platform's capabilities to other developers of the team. However, since in this context there is a single cross-functional team, it cannot be called a "platform team". We classified organization #I30 as a siloed structure, even with a platform team, since developers and the platform team have a conflicted relationship.

## 4 2020 An\_Empirical\_Taxonomy\_of\_DevOps\_in\_Practice

### 21 Quotations:

## 4:1 p 4 in 2020 An\_Empirical\_Taxonomy\_of\_DevOps\_in\_Practice

### Codes:

- automated application life-cycle management
- collaboration
- collaboration
- collaboration
- collaboration

#### Content:

Generally, participants described DevOps as better collaboration between developers and Ops teams. Some interviewees also described DevOps as end-to-end automation of the software development pipeline, providing better software quality, and creating seamless workflow of products at the shortest possible time.

### 4:2 p 4 in 2020 An\_Empirical\_Taxonomy\_of\_DevOps\_in\_Practice

#### Codes:

- collaboration
- collaboration
- collaboration
- collaboration
- containerization
- containerization
- containerization
- containerization
- devops (bridge) team
- devops (bridge) team
- devops (bridge) team
- devops (bridge) team

### Content:

The culture of collaboration between developers and Ops specialists was widely reported among interviewers. In FinCo1, the DevOps team assist developers reconfigure their codes for containerisation.

## 4:3 p 4 in 2020 An\_Empirical\_Taxonomy\_of\_DevOps\_in\_Practice

#### Codes:

- collaboration
- collaboration
- collaboration
- collaboration
- knowledge sharing
- knowledge sharing
- knowledge sharing
- knowledge sharing

#### Content:

Knowledge is shared and a mutual understanding of basic activities across the boundaries of teams, achieved through collaborative resolution of code-related challenges. Similarly, [Finco1 DOps2] mentioned knowledge flow from developers to DevOps Engineers:

## **■** 4:4 p 4 in 2020 An\_Empirical\_Taxonomy\_of\_DevOps\_in\_Practice

#### Codes:

- automated infrastructure management
- automated infrastructure management
- automated infrastructure management
- automated infrastructure management
- collaboration
- collaboration
- collaboration
- collaboration
- devops (bridge) team
- devops (bridge) team
- devops (bridge) team
- devops (bridge) team
- knowledge sharing
- knowledge sharing
- knowledge sharing
- responsibility/ownership sharing

### Content:

DevOps teams in these organisations understand the codes of developers and help out where necessary. Developers are also made aware of how the automated infrastructure works, though not directly involved in its creation or maintenance. According to some practitioners, a level of conffdence brought about by the basic understanding of other aspects of the process and familiarity with the other actors. Intra-team collaboration is reported as brainstorming and coding together when issues are encountered. Collaboration in FinCo2 involves the DevOps team creating users' stories from requirements, breaking them into manageable tasks and delegating these tasks to developers through Azure DevOps.

## 4:5 p 5 in 2020 An\_Empirical\_Taxonomy\_of\_DevOps\_in\_Practice

#### Codes:

- devops (bridge) team
- devops (bridge) team
- devops (bridge) team
- devops (bridge) team
- platform builder
- platform builder
- platform builder
- platform builder
- role definition/attributions
- role definition/attributions
- role definition/attributions
- role definition/attributions

#### Content:

Some of the interviewees had the job title of 'DevOps Engineer' and worked in distinct DevOps teams or departments. "We don't actually have developers in our team. So, in our case. . . it's just DevOps" [Finco1 DOps1]. They further described their team as "platform builders" for developers, "who support them and host their applications on our platform." Here, we see DevOps being presented as a job description, with DevOps Engineers responsible for carrying out "DevOps functions",

### 4:6 p 5 in 2020 An\_Empirical\_Taxonomy\_of\_DevOps\_in\_Practice

#### Codes:

- automated infrastructure management
- automated infrastructure management
- automated infrastructure management
- automated infrastructure management
- cross-functionality/skills
- cross-functionality/skills
- cross-functionality/skills
- cross-functionality/skills
- hybrid (on-premises & cloud)
- responsibility/ownership sharing
- responsibility/ownership sharing
- responsibility/ownership sharing
- · responsibility/ownership sharing

### Content:

Developers-Ops mode (shown in Fig. 2a) of DevOps implementation depicts the instance where senior developers performed automated infrastructure management alongside development activities in a hybrid cloud deployment environment. Here, senior developers were seen as the facilitators of DevOps practices.

## 4:7 p 5 in 2020 An\_Empirical\_Taxonomy\_of\_DevOps\_in\_Practice

#### Codes:

- automated application life-cycle management
- automated application life-cycle management
- automated application life-cycle management
- on-premises
- on-premises
- on-premises
- on-premises
- responsibility/ownership sharing
- responsibility/ownership sharing
- responsibility/ownership sharing
- responsibility/ownership sharing
- transfer of work between teams
- transfer of work between teams
- transfer of work between teams

#### Content:

The IT Ops team support the physical infrastructure and on-premises hosted application, while developers wrote application codes, deployed their codes through CI/CD pipelines, and managed applications themselves.

### 4:8 p 5 in 2020 An\_Empirical\_Taxonomy\_of\_DevOps\_in\_Practice

#### Codes:

- automated infrastructure management
- automated infrastructure management
- automated infrastructure management
- cloud
- cloud
- cloud
- cloud
- cross-functionality/skills
- cross-functionality/skills
- cross-functionality/skills
- cross-functionality/skills
- responsibility/ownership sharing
- responsibility/ownership sharing
- responsibility/ownership sharing
- responsibility/ownership sharing

#### Content:

The Developers-Outsourced Ops mode (shown in Fig. 2b) is similar to Developers-Ops mode described above. Senior developers also write infrastructure codes to create and manage deployment pipelines, the difference being that its deployment environment is cloud-based, eliminating the need for Ops experts.

## 4:9 p 5 in 2020 An\_Empirical\_Taxonomy\_of\_DevOps\_in\_Practice

### Codes:

- cloud
- devops (bridge) team
- devops (bridge) team
- devops (bridge) team
- devops (bridge) team
- platform builder
- platform builder
- platform builder
- platform builder

#### Content:

Fig. 2c depicts the Developers-DevOps mode where DevOps teams creates, deploys, and manages both the cloud infrastructure and deployment pipelines. Developers applications are also deployed and maintained by the DevOps team.

## **■** 4:10 p 5 in 2020 An\_Empirical\_Taxonomy\_of\_DevOps\_in\_Practice

#### Codes:

- devops (bridge) team
- devops (bridge) team
- devops (bridge) team
- devops (bridge) team
- responsibility/ownership sharing
- responsibility/ownership sharing
- responsibility/ownership sharing
- responsibility/ownership sharing
- transfer of work between teams

### Content:

Here, developers are not responsible for application deployment and management. Completed applications or features are handed over to the DevOps teams for deployment and management, who are the DevOps practices facilitators.

## **■ 4:11 p 5 in 2020 An\_Empirical\_Taxonomy\_of\_DevOps\_in\_Practice**

#### Codes:

- devops (bridge) team
- devops (bridge) team
- devops (bridge) team
- devops (bridge) team
- hybrid (on-premises & cloud)
- transfer of work between teams
- values & best practices

#### Content:

DevOps bridge team mode was the mode widely used in our study. This mode (shown in Fig. 2d) was found in hybrid environment of cloud and on premises deployment. Here, DevOps teams interface with both developers and IT Ops to drive the practices of DevOps like conffguration management, continuous integration and continuous delivery, automated testing, deployment, monitoring, and metrics collection.

### 4:12 p 5 in 2020 An\_Empirical\_Taxonomy\_of\_DevOps\_in\_Practice

#### Codes:

- devops (bridge) team
- devops (bridge) team
- devops (bridge) team
- devops (bridge) team
- · responsibility/ownership sharing
- responsibility/ownership sharing
- responsibility/ownership sharing
- responsibility/ownership sharing
- transfer of work between teams

### Content:

Developers provide business solution, leaving the creation, deployment, and management of both the cloud infrastructure, virtual systems, and deployment pipelines to the DevOps teams. These teams are provided services of on-premises infrastructure by the Ops team. Essentially, the DevOps teams are customers to the Ops team, and service providers to developers. It is important to note that in this approach to DevOps, everyone is responsible for their actions.

## 4:13 p 5 in 2020 An\_Empirical\_Taxonomy\_of\_DevOps\_in\_Practice

### Hyperlinks:

← continued by — 

4:15 p 5, these activities are solely the responsibility of the DevOps teams. in 2020 An\_Empirical\_Taxonomy\_of\_DevOps\_in\_Practice

#### Codes:

- automated application life-cycle management
- devops (bridge) team
- devops (bridge) team
- devops (bridge) team
- devops (bridge) team
- values & best practices

#### Content:

Using automation tools, DevOps engineers create pipelines to enable continuous practices such as continuous integration/continuous deployment, continuous testing etc.

### **4:14 p 5 in 2020 An\_Empirical\_Taxonomy\_of\_DevOps\_in\_Practice**

#### **Hyperlinks:**

← continued by — 

4:15 p 5, these activities are solely the responsibility of the DevOps teams. in 2020 An\_Empirical\_Taxonomy\_of\_DevOps\_in\_Practice

#### Codes:

- automated infrastructure management
- automated infrastructure management
- automated infrastructure management
- automated infrastructure management
- devops (bridge) team
- devops (bridge) team
- devops (bridge) team
- devops (bridge) team

### Content:

Scripts are created for conffguration management, and the deployment of infrastructure-as-code.

## **■** 4:15 p 5 in 2020 An Empirical Taxonomy of DevOps in Practice

### Hyperlinks:

— continued by  $\rightarrow$   $\bigcirc$  4:13 p5, Using automation tools, DevOps engineers create pipelines to enable continuous practices such as con... in 2020

An\_Empirical\_Taxonomy\_of\_DevOps\_in\_Practice

— continued by  $\rightarrow$   $\bigcirc$  4:14 p 5, Scripts are created for configuration management, and the deployment of infrastructure-as-code. in 2020

An\_Empirical\_Taxonomy\_of\_DevOps\_in\_Practice

#### Codes:

- devops (bridge) team
- devops (bridge) team
- devops (bridge) team
- devops (bridge) team
- responsibility/ownership sharing
- responsibility/ownership sharing

#### Content:

these activities are solely the responsibility of the DevOps teams.

### 4:16 p 5 in 2020 An\_Empirical\_Taxonomy\_of\_DevOps\_in\_Practice

### **Linked Memos:**

Organizational structures

#### Content:

Despite the seeming prevalence of bridge teams in the study, some interviewees thought it was not the right approach to DevOps implementation, as "there is still segregation between development and infrastructure in some way."

## **■ 4:17 p 7 in 2020 An\_Empirical\_Taxonomy\_of\_DevOps\_in\_Practice**

#### Codes:

- devops (bridge) team
- devops (bridge) team
- devops (bridge) team
- devops (bridge) team
- hybrid (on-premises & cloud)
- platform builder
- platform builder
- platform builder
- platform builder

#### Content:

The DevOps teams under study are be tasked with migration from existing platforms to either cloud based or an automated on premises environment, and its subsequent maintenance. Generally, they act as intermediary between IT Ops and developers, providing the means to an end in software development (SD) by creating automated pipelines on both physical and virtualized servers to enable continuous integration and continuous delivery.

## 4:18 p 7 in 2020 An\_Empirical\_Taxonomy\_of\_DevOps\_in\_Practice

#### Codes:

- devops (bridge) team
- devops (bridge) team
- devops (bridge) team
- devops (bridge) team
- responsibility/ownership sharing
- responsibility/ownership sharing
- responsibility/ownership sharing
- responsibility/ownership sharing
- transfer of work between teams

#### Content:

DevOps teams are responsible for all deployments. Developers hand over applications to these teams, who oversee the journey through the CI/CD pipeline. The teams also monitor the applications and function as ffrst line ofsupport.

## 4:19 p 7 in 2020 An\_Empirical\_Taxonomy\_of\_DevOps\_in\_Practice

#### Codes:

- devops (bridge) team
- devops (bridge) team
- devops (bridge) team
- devops (bridge) team
- responsibility/ownership sharing
- responsibility/ownership sharing
- responsibility/ownership sharing
- · responsibility/ownership sharing

#### Content:

Also, while the DevOps team works with the goal of giving developers the best tools to get their work done, they expect developers to take responsibility for their products. This suggests boundaries of responsibilities.

Another observed instance is the distinction between DevOps and IT Ops, clearly seen in the responsibility

## **■** 4:20 p 7 in 2020 An\_Empirical\_Taxonomy\_of\_DevOps\_in\_Practice

#### Codes:

- automated application life-cycle management
- devops (bridge) team
- devops (bridge) team

#### Content:

All deployments in the study, from developers and DevOps teams, go through automated pipelines.

## 4:21 p 7 in 2020 An\_Empirical\_Taxonomy\_of\_DevOps\_in\_Practice

### Codes:

- cross-functionality/skills
- cross-functionality/skills
- cross-functionality/skills
- cross-functionality/skills
- devops (bridge) team
- devops (bridge) team
- devops (bridge) team
- devops (bridge) team
- platform builder
- platform builder
- platform builder
- platform builder

#### Content:

our findings show DevOps being described by practitioners in the study as not just a culture and specific job description, but also distinct teams separate from both developers and IT Ops teams. Although members of these teams have backgrounds in either software development or IT Ops, the nomenclature "DevOps" now separates them from their original silos and classiffes them as a unique team of "platform builders"