A grounded theory of organizational structures for development and infrastructure professionals in software-producing organizations

Leonardo Leite[1], **Fabio Kon**[1], **Paulo Meirelles**[1]

[1]Department of Computer Science – Institute of Mathematics and Statistics (IME) – University of São Paulo (USP), São Paulo, Brazil

{leofl,kon,paulormm}@ime.usp.br

Abstract. *DevOps and continuous delivery have significantly impacted the organizational structures of development and infrastructure groups in software-producing organizations. Our research revealed the different options adopted by the software industry to organize such groups, unveiling why different organizations adopt distinct structures and how organizations handle the drawbacks of each structure. By interviewing 68 carefully-selected, skilled IT professionals and analyzing these conversations through a Grounded Theory process, we identified theoretical assertions (conditions, causes, reasons to avoid, consequences, and contingencies) related to each discovered structure (segregated departments, collaborating departments, API-mediated departments, and single department). We, then, offer a theory to explain organizational structures for development and infrastructure professionals. This theory supports practitioners and researchers in comprehending and discussing the DevOps phenomenon and its related issues; it also provides valuable input to practitioners' decision-making.*

Resumo. *DevOps e entrega contínua impactaram significativamente as estruturas organizacionais de grupos de desenvolvimento e infraestrutura em organizações produtoras de software. Nossa pesquisa revelou as diferentes opções adotadas pela indústria de software para organizar tais grupos, revelando porque diferentes organizações adotam estruturas distintas e como as organizações lidam com as desvantagens de cada estrutura. Ao entrevistar 68 qualificados profissionais de TI, que foram cuidadosamente selecionados, e analisar essas conversas por meio de um processo de Grounded Theory, identificamos afirmações teóricas (condições, causas, razões para evitar, consequências e contingências) relacionadas a cada estrutura descoberta (departamentos segregados, departamentos que colaboram, departamentos mediados por API e departamentos únicos). Oferecemos, então, uma teoria para explicar essas estruturas organizacionais para profissionais de desenvolvimento e infraestrutura. Essa teoria auxilia profissionais da indústria e pesquisadores na compreensão e discussão do fenômeno DevOps e suas questões relacionadas; ele também fornece informações valiosas para a tomada de decisões dos profissionais da indústria.*

Key words: DevOps, Grounded Theory, empirical software engineering, organizational structures, continuous delivery, software teams

1. Introduction

Here, we introduce our research by presenting its problem outline, research questions, and methodological approach.

1.1. Problem outline and research questions

To remain competitive, software-producing corporations seek to speed up their release processes. Organizations adopt continuous delivery practices in their quest to accelerate time-to-market and improve customer satisfaction. However, continuous delivery also comes with challenges, including profound impacts on various aspects of software engineering. With an automated deployment pipeline, one can, for example, question the role of an engineer responsible solely for new deployments. Since release activities involve many divisions of a company (e.g., development, operations, and business), adopting continuous delivery impacts an organization's structure. Therefore, organizations moving toward continuous delivery have not only to upgrade their software tooling arsenal but also to find ways to better shape and integrate their IT teams.

Given such recent transformations, there is a need for a better understanding of the organizational structures the software industry adopts for development and infrastructure employees. By organizational structure, we mean the differentiation (division of labor) and integration (interaction) [Oliveira 2012] of operational activities (application deployment, infrastructure setup, and service operation in run-time) between development and infrastructure groups. Empirical software engineering studies have focused on identifying the different organizational structures adopted in the industry to arrange development and infrastructure professionals [Nybom et al. 2016, Shahin et al. 2017, Fernandez et al. 2021, Macarthy and Bass 2020]. However, the literature available before our work did not reveal why different companies adopt different structures or how companies deal with the drawbacks of each structure.

This gap in the literature was inconvenient due to at least two crucial reasons: (1) organizations wishing to adopt continuous delivery can be disoriented regarding how to design their human resources structure toward this goal; (2) once a structure is chosen, the organization might be unaware of the consequences of this choice. The lack of a consolidated view of how the industry handles the outlined problem was also unfortunate. Decision-makers in software companies are highly interested in knowing "what other companies are doing" and "why are they doing it" to support their decisions.

In this thesis, we offer a theory to explain organizational structures for development and infrastructure professionals. We describe, in a taxonomy format, the structures currently adopted by software-producing organizations. Moreover, we explain why different organizations have different structures and how companies handle each structure's drawbacks. In other words, we present conditions, causes, reasons to avoid, consequences, and contingencies related to each structure.

In this way, this thesis answers the following research questions:

RQ1: *What are the organizational structures adopted by software-producing organizations to structure operational activities (application deployment, infrastructure setup, and service operation in run-time) among development and infrastructure* *groups?* **(RQ1.1)** *What are the properties of each of these organizational structures?* **(RQ1.2)** *Are some organizational structures more conducive to continuous delivery than others?*

RQ2: *Why do different software organizations adopt different organizational structures regarding development and infrastructure groups?* **(RQ2.1)** *How do organizations handle the drawbacks of each organizational structure?*

We investigated these questions in the context of software-producing organizations responsible for deploying the software they produce. For brevity, henceforward, we refer to them simply as "organizations" or "companies."

DevOps and continuous delivery are contextual factors that have impacted the software industry and are, therefore, part of the motivation for our research. Nevertheless, we clarify that the definitions and procedures of our research are not dependent on the notion of DevOps. We sought to understand the software production landscape regardless of whether companies claim to have adopted DevOps or not.

1.2. Research approach

Our research questions were not the fruit of inspiration solely. Our initial interest in this research endeavor was the DevOps topic. Thus, initially, we conducted a comprehensive literature review on DevOps. We wanted mainly to understand the DevOps field, with its concepts and challenges, and find research opportunities before working on an original contribution. Such a literature review employed some techniques of Systematic Literature Reviews (SLR) [Kitchenham and Charters 2007].

In particular, one of the DevOps challenges found in our literature review concerned the challenge of splitting roles across teams in the DevOps paradigm. To elaborate a taxonomy to describe the existing structures (**RQ1** and **RQ1.1**), we conducted semi-structured interviews with 37 IT professionals, each one belonging to a different organization. We analyzed these interviews by following Grounded Theory [Glaser and Strauss 1999], a methodology well-suited for generating theories. At the same time, at this point, we were eager to assess if a given structure could be somehow "better" than another. The dimension chosen for comparison was the delivery performance [Forsgren, N. et al. 2020] (**RQ1.2**).

After having these preliminary results, it was clear to us that there are distinct organizational structures in the industry used to organize development and infrastructure professionals. This situation led us to formulate **RQ2**: would there be legitimate reasons for different organizations to adopt different structures? Why do not all organizations adopt the "best" structure? Our intuition was that there was no "best" structure but different structures that were more suitable for different contexts. In this case, probably each structure would have its drawbacks. And if so, how would organizations handle such drawbacks (**RQ2.1**)? The search for such explanations also followed a Grounded Theory process.

To tackle **RQ2** and **RQ2.1**, we conducted semi-structured interviews with 31 carefully selected IT professionals working in 25 different companies, summing up 68 semi-structured interviews throughout our research. The explanations we offer about the structures are provided in the format of theoretical assertions: conditions, causes, reasons to avoid, consequences, and contingencies related to each discovered structure. This format corresponds to one of the theoretical coding families provided by the Grounded Theory methodology [Saldaña 2015]. This phase also had an additional goal: to interact with practitioners to somehow assess our taxonomy and improve it when necessary. During this process, we evolved the taxonomy by adding, removing, and renaming its high-level elements.

2. Results

Now, we highlight the significant results of our research effort, first talking about our survey of DevOps concepts and challenges, and after describing our original contribution, our grounded theory of organizational structures for development and infrastructure professionals in software-producing organizations.

2.1. Our literature review on DevOps concepts and challenges

Seeking to understand DevOps, we initiated our research at the beginning of 2018 by conducting a literature review searching for papers with the keyword “DevOps”. In this review, we thoroughly examined 50 scientific papers, which were carefully selected after we initially found almost 200 articles somehow related to DevOps. By analyzing these articles, we devised conceptual maps organizing DevOps concepts into four categories: process, people, delivery, and runtime. Figure 1 depicts the overall conceptual map; each quadrant of this map is unfolded in a broad map (these four maps are available in the thesis).

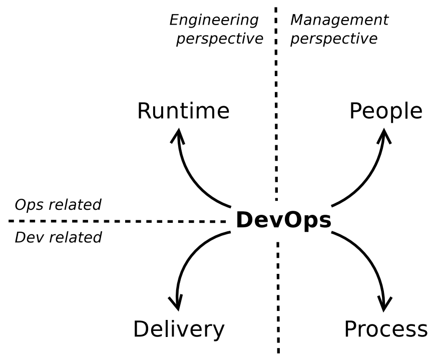

Figure 1. DevOps overall conceptual map

As part of our review, we also linked the detected concepts to the engineering’s and management’s perspectives; presented DevOps tools, classifying them and associating them to DevOps concepts; listed significant DevOps implications for engineers, managers, and researchers; and, finally, uncovered four fundamental DevOps challenges:

1. How to re-design systems toward continuous delivery, especially considering the relation between DevOps practices and microservices architecture.
2. How to organize the relations between development and infrastructure professionals to adopt DevOps, especially considering that mutual support between devs and ops is a very different thing from cross-functional teams.
3. How to assess the quality of DevOps practices in organizations, considering survey and system data and the pitfalls of any metric when used to promote or punish professionals.

4. How to qualify engineers for the DevOps practice, considering the laborious endeavor of professors preparing practical DevOps classes and evaluating students' assignments.

We selected the second listed challenge as our guide for this doctoral research. Based on our experience in Software Engineering and research, we considered the second challenge to be more manageable from an academic perspective, being more a sociological inquiry than an engineering endeavor.

2.2. Our theory on the organization of development and infrastructure professionals

Based on the careful analysis of the conducted interviews, we elaborated a theory describing the organizational structures used by the industry in the real world regarding how the work of developers and infrastructure engineers can be coordinated in the pursuit of continuous delivery. Such a theory encompasses a taxonomy describing different approaches for organizing development and infrastructure professionals (Figure 2) and attached theoretical assertions about each structure (conditions, causes, reasons to avoid, consequences, and contingencies). Some of the structures of our taxonomy have supplementary properties, which are optional structures. Now we briefly present these structures, their supplementary properties, and the conceived theoretical assertions.

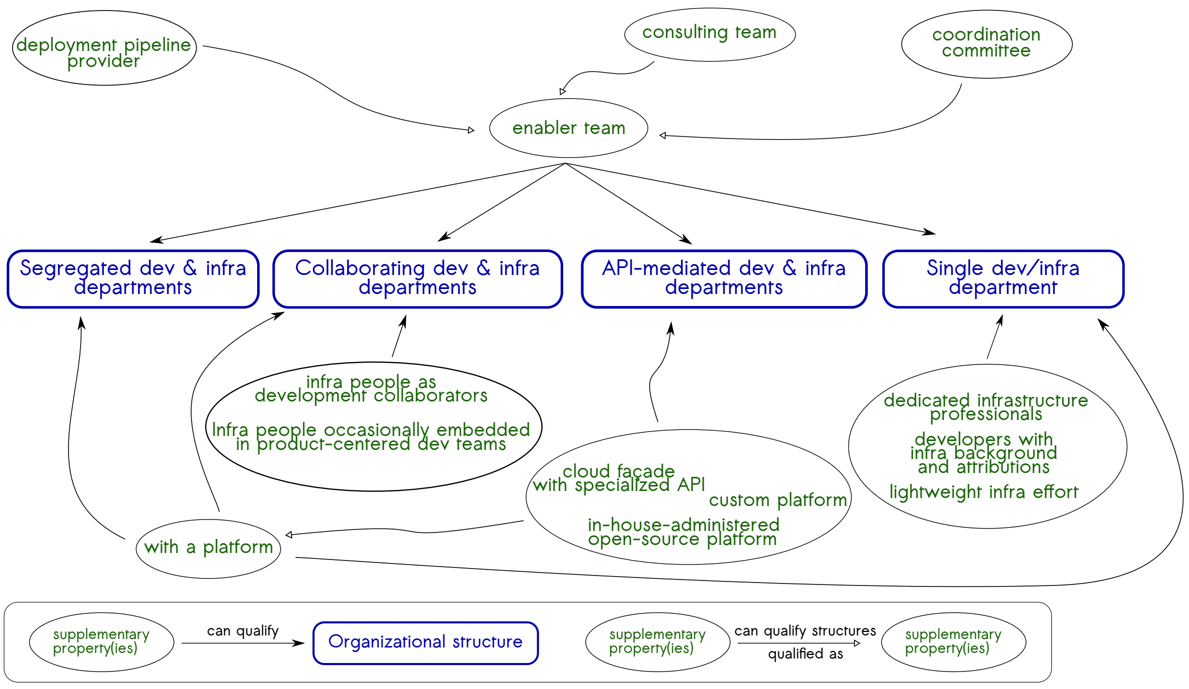
deployment pipeline provider
consulting team
coordination committee
enabler team
Segregated dev & infra departments
Collaborating dev & infra departments
API-mediated dev & infra departments
Single dev/infra department
infra people as development collaborators
Infra people occasionally embedded in product-centered dev teams
cloud facade with specialized API
custom platform
in-house-administered open-source platform
dedicated infrastructure professionals
developers with infra background and attributions
lightweight infra effort
with a platform
supplementary property(ies)
can qualify
Organizational structure
supplementary property(ies)
can qualify structures qualified as
supplementary property(ies)

Figure 2. A high-level view of our taxonomy

2.2.1. Segregated departments

Segregated, or siloed, departments is the "pre-DevOps" structure existing in large organizations, presenting limited collaboration among departments and barriers to continuous delivery. This structure is considered to be the problem that DevOps came to solve. Now we list our theoretical assertions about this structure.

Consequences of segregated departments

- Devs lack autonomy and depend on ops
- Low delivery performance (queues and delays)
- Friction and blaming games between devs and infra

2.2.2. Collaborating departments

This structure focuses on collaboration among developers and the infrastructure team. There is a culture of collaboration and communication among departments, which work in an aligned way. Non-functional requirements responsibilities, for example, are shared among developers and the infrastructure team. The supplementary properties and the theoretical assertions related to this structure are the following.

Supplementary properties of collaborating departments

Infra people as development collaborators: infrastructure engineers have advanced coding skills and contribute to the application code-base to improve the non-functional properties of the application.

Infra people occasionally embedded in product-centered dev teams: sometimes, collaboration occurs with an infrastructure professional spending some time in tight collaboration with a development team, especially at the beginning of a new project.

Conditions for collaborating departments

- Enough infra people to align with dev teams
- Top management support

Causes of collaborating departments

- In a non-large company / with few products, it is easier to be collaborative
- Trying to avoid the delivery bottleneck
- Bottom-up initiative with later top-management support

Consequences of collaborating departments

- Growing interaction inter-areas (e.g., knowledge sharing)
- Precarious collaboration (ops overloaded)
- Discomfort/frustration/friction/inefficiency with blurred responsibilities (people don't know what to do or what to expect from others)
- Waiting (hand-offs), infra still a bottleneck
- Automation supports collaboration

Contingencies for collaborating departments

- Giving more autonomy to devs (in staging or even production)

2.2.3. Single department

A single department takes responsibility both for software development and infrastructure management. It is more aligned with the Amazon motto "You built it, you run it," giving freedom to the team along with a great deal of responsibility. In particular, two challenges for this structure are enforcing corporate standards and guaranteeing every team has members with the necessary skills.

Supplementary properties of a single department

Dedicated infrastructure professionals: the team has at least one infrastructure specialist. It may be the case that the department has an infrastructure team dedicated to a group of developers. In this case, the difference between collaborating departments is the common hierarchy (same manager) for both development and infrastructure people.

Developers with infra background and attributions: the team has at least one developer with advanced infrastructure knowledge.

Lightweight infra effort: there is no need for advanced knowledge in infrastructure.

Conditions for a single department

- Enough ops for each dev team (associated with "dedicated infrastructure professionals")

Causes of a single department

- Startup scenario (small, young, weak infra scalability requirements, business focus, use of cloud services to limit costs)
- Cloud services decrease the need for infra & ops staff
- Delivery velocity, agility, critical project

Avoidance reasons for a single department

- Not suitable for applying corporate governance standards
- More costs: duplication of infra work among teams, high salaries for infra professionals, underused infra professionals

Consequences of a single department

- No [infra] defaults across teams: freedom, but possibly leading to duplication of efforts and high maintenance costs

Contingencies for a single department

- Improve infra skills in-house, including through tech talks (associated with "developers with infra background and attributions")

2.2.4. API-mediated departments

In this case, there is an infrastructure team (also called the platform team), and it provides highly automated infrastructure services ("the platform") abstracting the infrastructure to empower product teams. The existence of a delivery platform enables the product team to operate its own business services in production and minimizes the need for infrastructure specialists within the product team.

Supplementary properties of API-mediated departments

Cloud façade with specialized API: the company provides a platform consuming a public cloud (e.g., AWS) in a transparent way to developers.

In-house administered open-source platform: the organization manages an open-source platform, such as Rancher.

Custom platform: the organization builds its own platform due to the specific needs of the organization.

Conditions for API-mediated departments

- Medium to large-sized company
- Top-down initiatives/sponsorship
- Upfront investment
- Requires coding skills from infra people

Causes of API-mediated departments

- A delivery bottleneck in infra management
- Compatible with existing rigid structures (low impact on organogram) / Only a few people needed to form a platform team
- Fosters continuous delivery
- A hero or visionary (hero culture)
- Emerged as best solution; other initiatives not so fruitful
- Multiple products / multiple dev teams / multiple clients (requires high delivery performance)

Consequences of API-mediated departments

- Interaction (devs vs. platform team) to support devs, make things work, and demand new capabilities from the platform
- The platform provides common mechanisms (e.g., scaling, billing, observability, monitoring)
- Promotes continuous delivery, agility, and faster changes
- Devs responsible for infra architecture / concerns (e.g., NFR)
- The platform team provides consulting and documentation to devs
- Adding devs does not require adding [proportionally] more infra people
- Eliminated the previous bottleneck
- Small platform team (excellence center)
- High costs when using public clouds
- Devs skills are too focused on corporate needs, lacking base infra knowledge (bad for devs themselves, not for the company)
- The cost of managing the platform (even using open-source software) is high
- Risk: the platform is magic to devs; neglect quality because they trust too much in the platform, any problem they blame the platform and do not know what to do, even for simple problems or when the problem is in the application itself
- Devs possibly unable to understand the infra or to contribute to the platform

Contingencies for API-mediated departments

- Decide how much devs must be exposed to the infra internals (some places more, some places less)

2.2.5. Other supplementary properties

Enabler team: provides consulting and tools for product teams but does not own any service or infrastructure in the production environment. Another type of enabler team is a committee to coordinate the work of development and infrastructure groups.

With a platform: the organization possesses a platform but does not follow the patterns of human interaction and collaboration prescribed by API-mediated departments.

3. Conclusions

Our first contribution is the outline of DevOps concepts and challenges based on the literature. After, this doctoral thesis originally contributes to the area by presenting a systematically-derived theory of organizational structures, based on recent field observations and employing a well-accepted and systematic methodology. We describe in detail the structures currently adopted by the software industry to organize development and infrastructure professionals. In particular, it brings the following key benefits: *(i)* it helps practitioners to differentiate collaborating departments from a single department, which was traditionally blended under the term DevOps, and *(ii)* it highlights the API-mediated departments as a promising alternative – our taxonomy was the first empirically derived one to present such a structure. Therefore, it is also valuable that we provide empirical evidence that API-mediated departments lead to better delivery performance results.

Another significant contribution is providing an explanatory dimension for each structure presented in our taxonomy. We explicitly associated theoretical assertions with each structure. In this way, practitioners can understand why different organizations adopt different structures. They can also learn how organizations handle the drawbacks of each structure – or, at least, be aware of such drawbacks. Other researchers presenting similar taxonomies did not attribute such a level of explanation to their structures.

We also add as a secondary and methodological contribution the way we quantified theoretical saturation (the criteria of Grounded Theory to stop data collecting), providing more robust evidence about the saturation than is usual for software engineering studies. We hope our work will be an example in this aspect.

Importantly, the thesis also outlines the implications for multiple stakeholders. By increasing the awareness of organizational structures in the community, practitioners can make more informed decisions on structure selection and drawback handling. Moreover, we hope our theory can lead scholars to a deeper comprehension of the software production phenomenon so they can update software engineering classes and educational material considering the reality of the industry abstracted under our theory.

Our work has attracted the attention of practitioners and scholars. We already talked about our research in practitioners' podcasts and public and in-house events; the most notorious event was The Developer's Conference (TDC), in which we talked about platform teams in 2020 and 2022. We also lectured about our research and taught about DevOps in several educational institutions, such as the Paula Souza Center (Etec), the University of São Paulo (USP), the Federal University of Bahia (UFBA), and the Science Foundation Ireland Research Centre for Software (Lero). Besides featuring reputed Brazilian researchers in the DevOps field, the examining committee of this thesis also had the honor to count on international researchers from Spain and the USA. Our research group could then build a partnership with Spanish researchers working on similar projects [Fernandez et al. 2021]; such an international partnership, also involving an author of the notorious Teams Topologies book [Skelton and Pais 2019], has already resulted in a submitted paper about DevOps taxonomies [Alves et al. 2023].

Finally, about science communication, to provide an accessible reference for practitioners to follow, we keep a blog in which we report the results of our research[1]. In the

[1]`https://www.ime.usp.br/~leofl/devops/`

same expectation to amplify our impact in the practitioners' communities, we are now working on adapting the theses to the book format. We are writing this book in partnership with a publisher specialized in the market for Brazilian developers.

References

Alves, I., Pérez, J., Díaz, J., López-Fernández, D., Pais, M., Kon, F., and Rocha, C. (2023). Harmonizing DevOps taxonomies – theory operationalization and testing. Submitted to Empirical Software Engineering. https://arxiv.org/abs/2302.00033.

Fernandez, D. L., Diaz, J., Martin, J. G., Perez, J., and Prieto, A. G. (2021). DevOps team structures: Characterization and implications. *IEEE Transactions on Software Engineering*.

Forsgren, N. et al. (2020). A taxonomy of software delivery performance profiles: Investigating the effects of DevOps practices. In *Proceedings of The Americas Conference on Information Systems 2020*, AMCIS 2020, pages 1–6. AIS.

Glaser, B. and Strauss, A. (1999). *The discovery of grounded theory: strategies for qualitative research*. Aldine Transaction. Originally published in 1967.

Kitchenham, B. and Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering. Technical report, Keele Universit, University of Durham.

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

Nybom, K., Smeds, J., and Porres, I. (2016). On the impact of mixing responsibilities between devs and ops. In *International Conference on Agile Software Development*, XP 2016, pages 131–143. Springer International Publishing.

Oliveira, N. (2012). *Automated organizations: Development and structure of the modern business firm*. Physica-Verlag HD.

Saldaña, J. (2015). *The coding manual for qualitative researchers*. Sage.

Shahin, M., Zahedi, M., Babar, M. A., and 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*, EASE'17, pages 384–393. ACM.

Skelton, M. and Pais, M. (2019). *Team Topologies: Organizing business and technology teams for fast flow*. IT Revolution Press.