# Understanding context and forces for choosing organizational structures for continuous delivery

**Leonardo Leite (PhD student)**[1], **Fabio Kon (advisor)**[1], **Paulo Meirelles (co-advisor)**[1]

[1]PhD in Computer Science
Postgraduate Program in Computer Science (PPGCC-IME/USP)
Mathematics and Statistics Department (IME) – University of São Paulo (USP)
Rua do Matão, 1010 – CEP 05508-090 – São Paulo - SP – Brazil
Admission: Feb 2019 – Qualifying exam: Oct 2020 – Expected conclusion: Jun 2022

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

***Abstract.*** *In this research, we aim to understand the organizational structures adopted by software-producing organizations for managing IT technical teams in a continuous delivery context. Following Grounded Theory guidelines, we interviewed 46 IT professionals to investigate how organizations pursuing continuous delivery organize their development and operations teams. Among our results, we discovered four organizational structures: (1) siloed departments, (2) classical DevOps, (3) cross-functional teams, and (4) platform teams. After having discovered such structures and their properties, we describe, in this paper, our plans to better understand which contextual properties and forces lead an organization to adopt an organizational structure to the detriment of the other ones.*

***Resumo.*** *Nesta pesquisa, pretendemos entender as estruturas organizacionais adotadas por organizações produtoras de software para gerenciar equipes técnicas de TI em um contexto de entrega contínua. Seguindo as diretrizes da Grounded Theory, entrevistamos 46 profissionais de TI para investigar como as organizações que buscam a entrega contínua organizam suas equipes de desenvolvimento e operações. Entre nossos resultados, descobrimos quatro estruturas organizacionais: (1) departamentos em silos, (2) DevOps clássico, (3) equipes multifuncionais e (4) times de plataforma. Depois de descobrir essas estruturas e suas propriedades, descrevemos, neste artigo, nossos planos para entender melhor quais propriedades e forças contextuais levam uma organização a adotar uma estrutura organizacional em detrimento das demais.*

**Key words**: Continuous Delivery, Release Process, DevOps, Software Teams

**Related CBSoft symposia**: SBES

## 1. Problem characterization

To remain competitive, many software organizations seek to speed up their release processes. In this context, continuous delivery becomes decisive to accelerate time to market, improve customer satisfaction, and improved product quality [Chen 2015]. However, since continuous delivery impacts many divisions of a company (e.g., developers, operations, and business), organizations adopting it have to better shape and integrate their IT teams. Such integration can occur according to different patterns that we call *organizational structures*. However, there is no substantial literature tackling how organizations should structure their teams to excel in the context of continuous delivery. Although the existing literature presents some classifications for organizational structures [Nybom et al. 2016, Mann et al. 2018, Skelton and Pais 2013, Skelton and Pais 2019, Shahin et al. 2017], most of it presents sets of organizational structures without an empirical elaboration of how such sets were conceived.

To mitigate this gap, we have conducted an investigation to addresses the following research question (**RQ1**): *which organizational structures are software-producing organizations adopting for managing IT technical teams in a continuous delivery context?*

We answer this research question with a taxonomy, which is our emerging theory. A taxonomy is a classification system that groups similar instances to increase the cognitive efficiency of its users by enabling them to reason about classes instead of individual instances [Ralph 2019]. We applied Grounded Theory [Glaser and Strauss 1999], a well-suited methodology for generating taxonomies [Ralph 2019], to discover the existing organizational structures in the field by interviewing IT professionals.

Based on the analysis of the interviews data, we found four organizational structures: *i)* traditional *siloed departments*, with high impedance for cooperation among development and operations; *ii) classical DevOps*, focusing on the communication and collaboration among development and operations; *iii) cross-functional teams*, taking responsibility for both software development and infrastructure management; and *iv) platform teams*, exposing highly-automated infrastructure services to empower developers.

Each of these organizational structures has *core* and *supplementary properties*. An organization adhering a given structure will present, as consequence, most of the core properties associated with that structure. Supplementary properties support the explanation of more particular structural patterns, and an organization may or may not be associated with them. Figure 1 presents the discovered organizational structures, the primary elements of our taxonomy, alongside the supplementary properties, which qualify the elements pointed by the arrows. The circles group supplementary properties that can be equally applied for a given element.

The first version of our taxonomy, based on the first 27 interviews, was already presented at the International Conference on Software Engineering (ICSE) poster track [Leite et al. 2020a], and one of the found structures, platform teams, was described in detail in an ICSE workshop [Leite et al. 2020b]. Now, we have just submitted a new article[1] to the Information and Software Technology journal describing the current state of our taxonomy (Figure 1). The article is based on interviews with 46 practitioners working in 44 different organizations, and it fully explains how we conceived the taxonomy.

[1]The preprint is available on `https://arxiv.org/abs/2008.08652`.

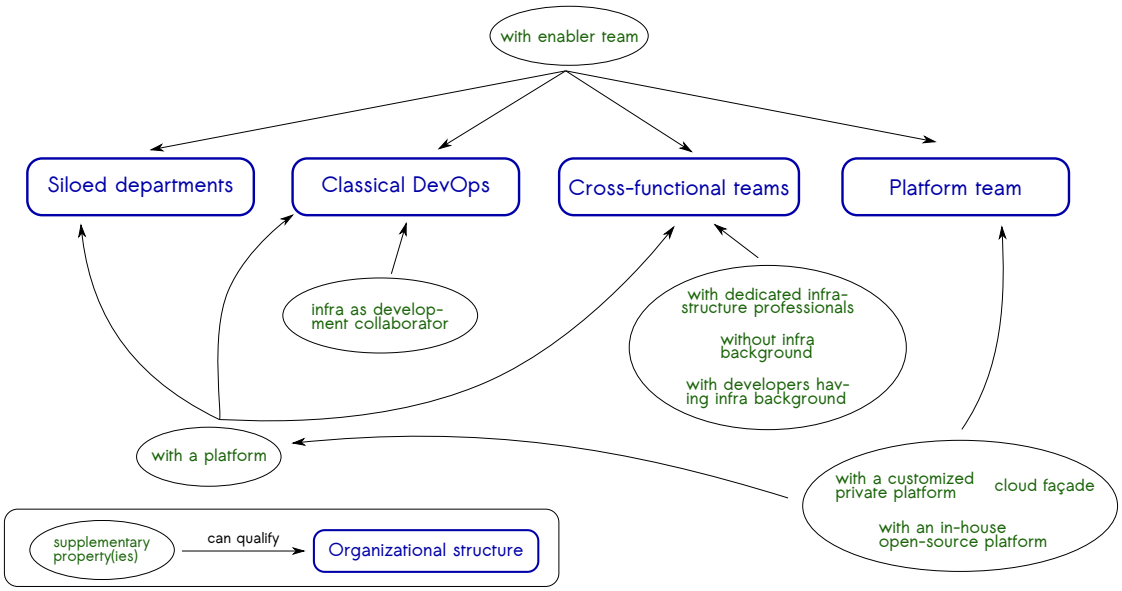


**Figure 1. High-level view of our taxonomy: discovered organizational structures and their supplementary properties**

Nonetheless, although we achieved success in producing a theory capable of describing organizational structures existing in the field, a related research question (**RQ2**) is still open: *which contextual properties and forces lead an organization to adopt an organizational structure to the detriment of the other ones?* In this article, we focus on describing our plans to answer **RQ2**.

## 2. Background

In our survey on the DevOps literature [Leite et al. 2019], published on the ACM Computing Surveys journal, we define DevOps as "*a collaborative and multidisciplinary effort within an organization to automate continuous delivery of new software versions, while guaranteeing their correctness and reliability*."

The existence of distinct silos for operations and development is still common in industry: development teams continuously seek to push new versions into production, while operations staff attempt to block these changes to maintain software stability and other non-functional concerns. In practice, this arrangement results in long delays between code updates and deployment, besides problem-solving ineffectively lead by blaming games. From an organizational perspective, the DevOps movement promotes closer collaboration between developers and operators to overcome such problems.

In a more technical perspective, the core practice promoted by the DevOps movement is continuous delivery: any software version committed to the repository must be a production-candidate version; after passing through stages, such as compilation and automated tests, the software is sent to production by the press of a button [Humble and Farley 2010]. DevOps initiatives have also approached the improvement of software run-time properties, such as performance, scalability, availability, and resilience [Beyer et al. 2016, Basiri et al. 2016].

Although our DevOps definition points to different departments within an organi-

zation helping each other, some approaches advocate bringing developers and operators together in cross-functional teams [Gray 2006]. Given this dichotomy, in our DevOps survey [Leite et al. 2019], we highlight the lack of empirical studies about how to structure an organization in the context of DevOps adoption.

Another relevant concept for our research is delivery performance, a construct that combines three metrics: frequency of deployment, time from commit to production, and mean time to recovery [Forsgren et al. 2018]. Delivery performance also correlates to the capability of achieving commercial and noncommercial goals of the organization. We used this construct in our research as an indication of how successful the organization has been in adopting continuous delivery. Originally, high performers organizations have multiple deployments per day, commits taking less than 1 hour to reach production, and incidents repaired in less than one hour [Forsgren et al. 2018]. We also considered an organization as a high performer if it violates at most one high-performance threshold by only one point in the scale adopted for the metric.

## 3. Methodology

For answering **RQ1**, we employed Grounded Theory [Glaser and Strauss 1999]. We started with no predefined hypotheses and built a taxonomy grounded on data we collected from interviews. Having now theoretical support, we will employ the Case Study methodology for answering **RQ2**. A case study is "*an empirical inquiry that investigates a contemporary phenomenon in depth and within its real-life context*" [Yin 2009]. By following Yin's guidelines, we avoid the lack of rigor from which case studies are regularly criticized [Yin 2009].

Yin characterizes the suitable scenario for applying Case Study: *i)* the research poses "how" or "why" questions; *ii)* the researcher has little control over the studied events; and *iii)* the focus is on contemporary phenomenons of real-life. Items *ii* and *iii* hold since we are interested in real-life properties of software-producing organizations, while we have no power to control how companies are organized. About item *i*, **RQ2** could be rephrased as "***why*** *different organizations choose (or should choose) different organizational structures?*". Therefore the requirements for applying the Case Study methodology hold.

A frequent critic of case studies is that their results are non-generalizable since the study is based on a small sample of the studied phenomenon. About this, Yin argues that case studies are generalizable to theoretical propositions and not to populations or universes. In this way, case studies must be strongly linked to an in-development theory, and include a set of predefined hypotheses about the studied phenomenon. Such hypotheses will support several choices in designing the case study and support the generalization of results. Another important aspect of formulating early theoretical propositions is that it allows us to formulate rival explanations, i.e., alternative narratives that could be applied in case our theory is wrong. According to Yin, such rival explanations play a relevant role in the quality of the case study. In Section 5, we present our initial set of hypotheses and rival explanations for our case studies.

Case studies must define the unit of analysis. It must not be an abstract concept, such as "neighboring," but rather a concrete entity as, for example, a specific neighborhood. Our unit of analysis are the interaction dynamics (or their absence) among develop-

ment and operations groups according to its current status. This definition sets spacial and temporal boundaries of our cases, as recommended by Yin. Although one could point the whole organization as an option of unit of analysis, we avoided this option since different teams within an organization can interact according to different patterns of interaction.

Case studies can present a single case or multiple cases. Due to theoretical necessities, we will conduct a multiple case study for analyzing at least one case for each one of these organizational structures: classical DevOps, cross-functional teams, and platform teams. There is no need to conduct a case of a pure siloed-departments organization since it is the "pre-DevOps structure" and our theory is intended to support DevOps adoption. Moreover, we will choose organizations that present high-delivery performance [Forsgren et al. 2018], so they are more suitable as sources to the formulation of a theory able to help other organizations to achieve high-delivery performance. For choosing organizations that are high performers and adopt defined structures, we will choose organizations already interviewed by us.

Yin poses that each individual case of a multiple case study can be "holistic" or "embedded." The embedded case can be split into more units of analysis; in such a way, the analysis of each unit is condensed within the case, and the results per case are compared. On the other hand, in a holistic case, there are not sub-units of analysis for each individual case, so each case is analyzed holistically. In our case, even considering that, within a single organization, we can interview different people of different roles (e.g., developers, infrastructure specialists, and managers), we cannot say each interview is a unit of analysis since we are interested in the interactions among these actors. In this way, we design our case study as a holistic multiple case study.

Yin highlights the importance of using multiple sources of evidence for triangulation in case studies. Accordingly, the author lists the following sources of evidence: documentation, archival records, interviews, direct observations, participant-observation, physical artifacts. Our primary and most obvious sources of evidence are interviews. The main source of triangulation will be interviewing at least two persons in each selected organization: a dev-profile person and an ops-profile person. If possible, we will try to reach four interviewees per organization: a dev engineer, a dev manager, an ops engineer, and an ops manager. As complementary sources of evidence, we can take official documentation, such as the chart of the company and hiring ads, and delivery metrics. In Section 5, we present our initial plans for data collection.

## 4. Method for evaluating results

Yin points out four quality criteria commonly adopted to assess case study design and tactics for meeting such criteria.

*Construct validity* is about taking the right measures for the concepts under study. The strategies to coping with the threat of missing construct validity are three: having multiple sources of evidence; establishing a chain of evidence; and having research participants reviewing the final report. We plan to adopt all these measures.

*Internal validity* concerns the validity of implications, i.e., whether relations like $x \Rightarrow y$ are indeed causal relations and not only spurious correlations. For this, the four handling strategies are pattern marching, explanation by building, addressing rival expla-

nations, and using logic models. From these, the most suitable for us are explanation by building and addressing rival explanations.

*External validity* is about reasoning whether the study results are generalizable beyond the presented case. For handling this issue, the researcher can rely on theory or replication. In our study, we rely mainly on analytical generalization regarding the careful construction of our emerging theory, already grounded on 46 interviews and feedback from participants. Replication is left as future research that can reinforce or change our theory.

*Reliability* relates to reproducibility and can be handled by the definition of a case study protocol and the maintenance of a study database, which we will follow.

## 5. Current state of work

Based on data retrieved from interviews with IT professionals, we developed a taxonomy presenting organizational structures that describe patterns of interactions among development and operations professionals. This taxonomy constitutes our emerging theory. Based on this emerging theory, we can now define hypotheses that will conduct the research to answer **RQ2**, which is about understanding the context and forces that lead different organizations to choose different organizational structures.

We hope to hold discussions with other researchers to define and refine our hypotheses. We expect particularly to discuss them with colleagues with a more "ops view of the world," given the "dev view of the world" of this PhD student. Nonetheless, based on the so-far conducted observations, we present here our initial set of hypotheses:

- Siloed departments can be a valid option when the organization understands it does not need to achieve high delivery performance.
- Although it is not promising for achieving high delivery performance, classical DevOps is an adequate first step for large siloed organizations to undertake a DevOps transformation.
- Classical DevOps is preferable when the application has advanced non-functional requirements or a very large scale since, in this case, it makes sense to have a dedicated operations group taking care of the application.
- Cross-functional teams are a natural path for small organizations since there is no much sense in creating multiple departments in this context.
- Cross-functional teams can be more easily applied to large an organization if it produces many different and unrelated applications.
- Platform teams are not suitable for small organizations since there are not specialized people enough to form a platform team.
- The company domain has no impact on the choice of organizational structure.
- A healthy culture and proper engineering practices precede the choice of organizational structure for the achievement of high delivery performance.

Yin also indicates as important the definition of rival explanations. Our start point for such alternative hypotheses, based on related work and our own data, are:

- For any company, cross-functional teams should always be aimed since reputed practitioners recommend it.

- Since the platform team is the most promising structure to achieve high delivery performance, it should always be the preferred structure.

The essence of the data collection procedure will be asking each interviewee why the organization adopted its organizational structure. According to the situation, we will also ask why it did not adopt another organizational structure or why it should adopt another structure. We also expect that discussions with other researchers can refine such procedures.

## 6. Expected contributions

We already have published the following articles with the corresponding contributions:

- "A survey of DevOps concepts and challenges," published on the ACM Computing Survey journal [Leite et al. 2019]. It presents: DevOps concepts organized in conceptual maps; DevOps tools associated to DevOps concepts; DevOps implications for engineers, managers, and researchers; and unresolved DevOps challenges.
- "Building a theory of software teams organization in a continuous delivery context," published as an extended abstract for the $42^{nd}$ International Conference on Software Engineering (ICSE 2020) poster track [Leite et al. 2020a]. It briefly presents the organizational structures of our taxonomy.
- "Platform teams: An organizational structure for continuous delivery," published at the 6th International Workshop on Rapid Continuous Software Engineering (RCoSE 2020), held in conjunction with ICSE 2020 [Leite et al. 2020b]. It describes in detail the properties of platform teams, the organizational structure of our taxonomy with the best delivery performance results.

Now, we have just submitted an article to the Information and Software Technology journal describing the current version of our taxonomy, which provides the academic contribution of knowing and understanding the existing organizational structures to support the adoption of continuous delivery. This knowing and understanding, consolidated in a taxonomy, provides benefits to the industry: after choosing and adopting an organizational structure, for whatever reason, the organization can be fully aware of the consequences of this choice.

Our next expected academic contribution is to understand the contextual properties and the forces that lead an organization to adopt a structure and not others. We expect that such discoveries enable us to describe the organizational structures in a language of patterns [Meszaros and Doble 1997]. By answering our second research question, we contribute to the industry by providing a better base for people within the organizations to discuss organizational changes towards continuous delivery.

Thus, we hope this doctoral research to provide as contribution a theory able to support the discussion of organizational changes in the continuous delivery context. We also clarify that we do not expect to conduct theory validation, which would require another extensive research by its own.

## 7. Comparison with related works

The literature about the inter-team arrangements for managing IT infrastructure in a continuous delivery context is still limited. Most of the available works present

sets of organizational structures without an empirical elaboration of how such sets were conceived [Humble and Molesky 2011, Nybom et al. 2016, Mann et al. 2018, Skelton and Pais 2013, Skelton and Pais 2019]. The *Team Topologies* book shows evidence that the proposed topologies emerged from field observations, but lacking a scientific methodology. An exception is the work of Shain *et al*, which follows scientific guidelines to discover in the field DevOps organizational structures [Shahin et al. 2017]; they also try to establish rationales for adopting their structures.

Shahin *et al.* conducted semi-structured interviews in 19 organizations and surveyed 93 practitioners to empirically investigate how development and operations teams are organized in the software industry for adopting continuous delivery practices. They found four types of team structures: *i) separate Dev and Ops teams with higher collaboration*, *ii) separate Dev and Ops teams with facilitator(s) in the middle*; *iii) small Ops team with more responsibilities for Dev team*, and *iv) no visible Ops team*. They also explore the size of the companies adopting each structure. They found that structure *i* is mainly adopted by large organizations, while structure *iv* was observed mainly in small ones. Our data corroborate these findings: classical DevOps was less observed in small organizations, while cross-functional teams were not prevalent in large organizations. This similarity of independent results is a relevant outcome to the field. So these considerations lay a good base for our formulation of hypotheses for the case studies. However, other factors may be involved in adopting an organizational structure.

Finally, in another work, Shahin and Babar recommend adding operations specialists to the teams [Shahin and Babar 2020], which favors cross-functional teams with dedicated infra professionals. Srivastava *et al.* also consider cross-functional teams as a positive factor, favoring developer velocity [Srivastava et al. 2020]. On the other hand, we expect to understand in which conditions cross-functional teams are indeed preferable to other organizational structures. Nonetheless, such a proposition of always favoring cross-functional teams can be the base of rival explanations for our case studies.

## Acknowledgements

We thank the support of the Brazilian Federal Data Processing Service (Serpro), CNPq proc. 465446/2014-0, CAPES – Finance Code 001, and FAPESP proc. 14/50937-1 and 15/24485-9.

## References

Basiri, A., Behnam, N., de Rooij, R., Hochstein, L., Kosewski, L., Reynolds, J., and Rosenthal, C. (2016). Chaos engineering. *IEEE Software*, 33(3):35–41.

Beyer, B., Jones, C., Petoff, J., and Murphy, N. R. (2016). *Site Reliability Engineering: How Google Runs Production Systems*. O'Reilly Media.

Chen, L. (2015). Continuous delivery: Huge benefits, but challenges too. *IEEE Software*, 32(2):50–54.

Forsgren, N., Humble, J., and Kim, G. (2018). Measuring performance. In *Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations*. IT Revolution Press.

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

Gray, J. (2006). A conversation with werner vogels. *ACM Queue*, 4(4):14–22.

Humble, J. and Farley, D. (2010). *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. Addison-Wesley Professional.

Humble, J. and Molesky, J. (2011). Why enterprises must adopt devops to enable continuous delivery. *Cutter IT Journal*, 24(8):6–12.

Leite, L., Kon, F., Pinto, G., and Meirelles, P. (2020a). Building a theory of software teams organization in a continuous delivery context. In *42nd International Conference on Software Engineering Companion*, ICSE '20 Companion, pages 294–295.

Leite, L., Kon, F., Pinto, G., and Meirelles, P. (2020b). Platform teams: An organizational structure for continuous delivery. In *IEEE/ACM 42nd International Conference on Software Engineering Workshops*, ICSEW'20, pages 505–511.

Leite, L., Rocha, C., Kon, F., Milojicic, D., and Meirelles, P. (2019). A survey of DevOps concepts and challenges. *ACM Computing Surveys*, 52(6):127:1–127:35.

Mann, A., Brown, M. S. A., and Kersten, N. (2018). 2018 State of DevOps Report. `https://puppet.com/resources/whitepaper/2018-state-of-devops-report`, accessed on Jul 2019.

Meszaros, G. and Doble, J. (1997). A pattern language for pattern writing. In *Proceedings of International Conference on Pattern languages of program design (1997)*, volume 131, page 164.

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.

Ralph, P. (2019). Toward methodological guidelines for process theories and taxonomies in software engineering. *IEEE Transactions on Software Engineering*, 45(7):712–735.

Shahin, M. and Babar, M. A. (2020). On the role of software architecture in devops transformation: An industrial case study. Accepted in ICSSP 2020. `https://arxiv.org/abs/2003.06108`, accessed on May 2020.

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. (2013). Devops topologies. `https://web.devopstopologies.com/`, accessed on Jul 2019.

Skelton, M. and Pais, M. (2019). *Team Topologies: Organizing Business and Technology Teams for Fast Flow*. IT Revolution Press.

Srivastava, S., Trehan, K., Wagle, D., and Wang, J. (2020). Developer velocity: How software excellence fuels business performance. `https://mck.co/2xENHUT`, access on Aug. 2020.

Yin, R. K. (2009). *Case Study Research, Design and Methods*. SAGE Publications, 4th edition.