Building a Theory of Software Teams Organization in a Continuous Delivery Context

Leonardo Leite, Fabio Kon  
{leofl,kon}@ime.usp.br  
University of São Paulo, Brazil

Gustavo Pinto  
gpinto@ufpa.br  
Federal University of Pará, Brazil

Paulo Meirelles  
paulo.meirelles@unifesp.br  
Federal University of São Paulo, Brazil

ABSTRACT

Based on Grounded Theory guidelines, we interviewed 27 IT professionals to investigate how organizations pursuing continuous delivery should organize their development and operations teams. In this paper, we present the discovered organizational structures: (1) siloed departments, (2) classical DevOps, (3) cross-functional teams, and (4) platform teams.

CCS CONCEPTS

- Software and its engineering → Software development process management; Programming teams; Software post-development issues.

KEYWORDS

Continuous Delivery, Release Process, DevOps, Software Teams

ACM Reference Format:


2 APPROACH

For tackling our research question, we are building a theory in the taxonomy form. Taxonomies, common on software engineering research, are classification systems that group similar instances to increase the cognitive efficiency of its users by enabling them to reason about classes instead of individual instances [7]. If the taxonomy provides explanation, it can be considered a theory for understanding, a system of ideas for making sense of what exists or is happening in a domain [7].

In this work we applied Grounded Theory [2], a well-suited methodology for generating taxonomies [7] and a widely-used research approach in software engineering [3, 4, 9], to discover the existing organizational structures in the field. Grounded Theory is adequate for our purposes since it is suitable for questions like “what’s going on here?” – we want to know what is going on software-producing organizations – and useful to construct a relevant conceptual and theoretical foundation for the field [9].

We conducted interviews with objective questions about the working day-to-day of participants to find out the underlying organizational structures. Initially, we had brainstorming conversations with seven specialists, who helped us to better understand the relevance of the problem and to shape the questions to be asked in follow-up interviews. We then conducted 20 semi-structured interviews with IT professionals. The only requirement was that the company should have a continuous delivery process or at least an initiative toward it.

We employed several strategies to foster diversity and to enhance comparison possibilities in our sample. We choose a broad range of organization and interviewee profiles: different company sizes (7 companies with more than 1,000 employees), domains, countries (Brazil, USA, Germany, and France), genders (7 women), experience levels (11 interviewees graduated more than ten years ago), and roles (developers, managers, infrastructure specialists, and even a
We found four organizational structures:

i) We classified eight interviewees contexts as traditional siloed departments, with limited collaboration among departments and barriers for continuous deployment. Some typical characteristics: developers and operators have well-defined and different roles; developers have a minimal vision of what happens in production; monitoring and handling incidents are mostly done by the infrastructure team; developers often neglect non-functional requirements (NFR); security can be seen as an infrastructure concern only; DevOps initiatives are centered on adopting continuous integration tools rather than improving collaboration among silos; as a consequence, communication and collaboration among teams are hard.

ii) We classified six interviewees contexts as having a classical DevOps structure, with intense collaboration among developers and the infrastructure team. Some typical characteristics: roles remain well-defined, although developers and operators are closer (e.g., for database management, infrastructure staff creates and tunes the database, whereas developers write queries and manage the schema), which fosters a culture of collaboration; usually, there are no conflicts regarding who is responsible for each task; DevOps is achieved through a delivery pipeline; developers and the infrastructure team share NFR responsibilities; the infrastructure staff is still in the front line of tracking monitoring and incident handling; success of classical DevOps requires strong alignment among departments.

iii) We classified three interviewees contexts as possessing cross-functional teams, with self-sufficient teams having both development and operations skills. Some typical characteristics are: a single team encompasses both developers and infrastructure specialists to take total responsibility for the life cycle of a set of services; this structure is the one that most supports communication and collaboration among people with different skills; everyone in the team can be assigned to incident handling; the challenge here is to guarantee that each unit has all the necessary skills.

iv) We classified three interviewees contexts as presenting platform teams, with the infrastructure team providing highly-automated infrastructure services to empower product teams. Some typical characteristics are: the existence of a delivery platform minimizes the need for product teams having infrastructure specialists; product teams become decoupled from the members of the platform team; usually, the communication among the development team and the platform team happens when infrastructure members provide consulting for developers; the product team is the first one to be called when there is an incident; the infrastructure people are escalated if the problem is related to some infrastructure service; although the product team becomes fully responsible for NFRs of its services, it is not a significant burden, since the platform abstracts away the underlying infrastructure and handles several NFR concerns.

4 CONCLUSION

Our work presents an initial taxonomy of organizational structures, based on recent observations from the field employing a well-accepted methodology. In particular, our proposed taxonomy points to the benefits of i) helping practitioners to differentiate classical DevOps from cross-functional teams, which were traditionally blended under the term DevOps, and ii) highlighting the platform team as a distinctive choice for organizations. We expect the under-development taxonomy to be applicable to empowering practitioners to discuss the current situation of organizations, supporting decisions on structural changes; another application would be supporting, for example, engineers in job interviews to evaluate the suitability of working for a given company.

ACKNOWLEDGMENTS

This research was supported by CNPq (proc. 465446/2014-0, 309032/2019-9, and 406308/2016-0), FAPESP proc. 15/24485-9, and FAPESPA.

REFERENCES