Leveraging DevOps for mission critical software

Emergent Research Forum (ERF)

Michael Gall
Grenoble Ecole de Management
michael.gall@grenoble-em.com

Federico Pigni
Grenoble Ecole de Management
federico.pigni@grenoble-em.com

Abstract

Using agile development and operations (DevOps) practices to deliver software as a service (SaaS) is often considered unsuitable for large scale, distributed software projects delivering mission critical software. Although there is an increasing call for agile delivery to be able to react on ever changing business requirements, there is little empirical research that shows when and how DevOps practices can be adopted successfully within an organization to deliver mission critical software. Based on the preliminary results of a case study at a multinational software company, we identify critical issues and suggest a method to overcome them, so that adopting DevOps practices for mission critical software is feasible.

Keywords

DevOps, agile software development, Software as a Service

Introduction

Software as a Service (SaaS) has imposed itself as the major delivery mode in the software industry requiring developers to transition from established software development methodologies towards agile ones. SaaS solutions go beyond the development of the software and include the delivery and operational aspect of the information system. In recent years, development and operations (DevOps) has emerged as a practice in the context of agile software development. DevOps comprises the development and operation of information systems (Ghantous and Gill 2017). DevOps claims to enable shorter response times when it comes to the delivery of features or bug fixes utilizing a continuous integration and continuous deployment (CI/CD) pipeline (Humble and Farley 2010). Despite benefits of onboarding agile practices, existing organizational and development practices often hinder the adoption (Erich et al. 2017). To overcome the resulting resistance to change, organizations often resorted to two separate but coherent styles of work (Gartner 2015) depending on the nature of the project. One mode applied for the development of mission critical software following rigid waterfall methodology. The second mode centered on rapid software development utilizing agile software development for non-critical software. Mission critical software is defined as business applications, excluding email, that would bring an organization to a stop if it were not running (Poba-Nzaou and Uwizeyemungu 2013). Mission critical software systems are not necessarily large or complex, however, due to their nature these systems traditionally follow more rigid processes regarding risk, compliance and security (Spurrier and Topi 2017). Non-critical systems, complementing the mission critical systems, do not directly impact the organization's ability to achieve its mission and goals (Poba-Nzaou and Uwizeyemungu 2013). At its core, this research is contesting the prescription of bimodal IT and analyzes the application of a DevOps model for mission critical software. Successful adoption of DevOps has proven to foster innovation and enhance the speed of IT product delivery (Kranz et al. 2016) which is equally important for both non mission critical software and for mission critical software, hence gaining a deeper understanding of DevOps adoption for mission critical software is crucial. Accordingly, we put forth the following research questions:

What are the factors affecting the adoption of DevOps within an organization for the development of mission critical software?

To answer this research question, we collected mixed data (polls, interviews, and observation) in a software development project for mission critical software at a multinational software company where DevOps practices are adopted. After a brief introduction of the theoretical concept and a review of the relevant literature, we describe the methodology and provide insights into data collection process followed by a presentation and discussion of the initial findings. Finally, we summarize the findings and highlight the contribution and limitations as well as ongoing research based on this case study.

Theoretical Background

DevOps is based on two core principles: (1) collaboration between development and operations; (2) use of agile principles and automation tools for the continuous delivery of software as a service (Humble and Farley 2010). DevOps aligns the development of software with the deployment of software into production (Fitzgerald and Stol 2017). In this model, a DevOps team represents a cross-functional team which is a unit of an organization and possesses the end to end responsibility for the service that is being developed, deployed, monitored and operated in various different environments (Mullaguru 2015). DevOps teams achieve agility by replacing traditional organizational structures and approaches that try to achieve economies of scale through a strict division of labor between organizational units (Davis and Daniels 2016).

Relevant Literature

The DevOps concept is not clearly defined and currently twenty different DevOps practices that detail eight conceptual elements can be found in existing literature, which makes it challenging for organizations to effectively adopt DevOps at scale (Ghantous and Gill 2017). Several sources discuss the transformation of an organization into one running an agile software development process delivering a modern service. Gruver and Mouser (2015) describe such a transformation in HP, however the software that is being transformed is a non-mission critical system. Shropshire and Sweeney (2017) underline the need for further research on how DevOps impacts the workforce moral. Erich, Amrit and Daneva (2017) highlight several issues companies are facing when adopting DevOps practices. The degree of which new practices are assimilated within organizations is then described along six stages (Initiation; Adoption; Adaption; Acceptance; Routinization; Infusion) (Gallivan 2001). Only limited research has been conducted to investigate the acceptance and later stages of the assimilation of DevOps within organizations (Wiedemann 2017). Lwakatare, Kuvaja and Oivo (2015) investigate the elements that characterize DevOps and identify four main dimensions: collaboration, automation, measurement, and monitoring. Lwakatare et al (2015) call out a research gap in regards to the common understanding of DevOps and call out the need for further investigation. Horlach, Drews, Schirmer and Böhmann (2017) highlight that for the development of non-critical systems usually the DevOps methodology is applied, thus remarking the need of a deeper understanding of both the DevOps methodology and its adoption.

Research Methodology

This explorative study examines the adoption of DevOps practices for mission critical software and its challenges. To answer the research question, we collected mixed data (structured polls, semi-structured interviews, and unstructured observations) in a mission critical software development project. The data collection started in October 2017 and ended in the beginning of February 2018.

Case Description

The project object of this case study is the delivery of an application development platform (ADP) as a SaaS solution. Looking at the ADP in more detail, it consists of 4 areas: (C1) a monolithic server, (C2) a runtime user interface (UI), (C3) a design time UI and (C4) satellite services providing additional functionality that is not mission critical. This structure is reflected in newly formed DevOps teams containing overall 550 software engineers that are delivering this platform. Approximately half of the engineers in these teams had been using DevOps practices over several years. However, these developers are new to the project and are therefore unfamiliar with its code base. The other half of the engineers originated from the product group who previously developed the first version of the ADP and know the codebase very well. However, these engineers have been used to delivering software following a traditional waterfall-based model (Sommerville 2015) and have been completely shielded from operational aspects of running the ADP as a service.

Data Collection Approach

To analyze challenges the development team had been facing while applying DevOps practices, we applied a mixed method approach in the case study. The project timeline and at which point during the project data was collected is visualized in Figure 1.

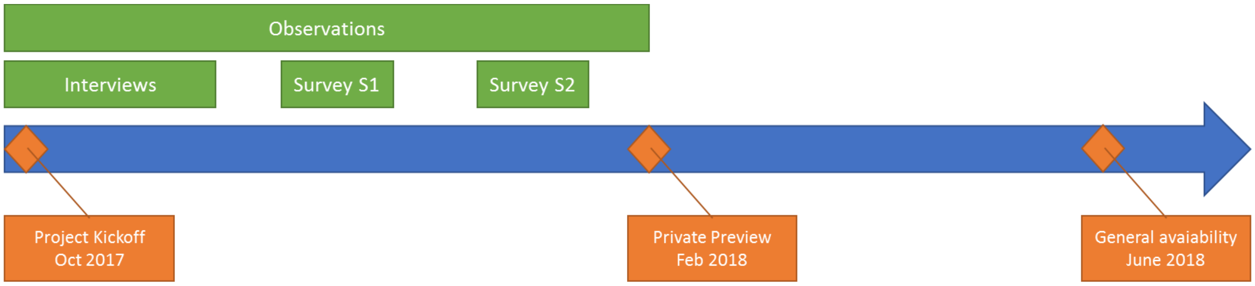

Figure 1: Project timeline

For the first component, we collected data to understand the context of the project, current processes, and work practices. Over a span of two weeks, we conducted short interviews with 26 software engineers (SE), 4 engineering managers (EM) and 2 software architects (SA). The participants have been selected based on seniority and the area they are working on to obtain data from both senior and junior developers from all areas of the project. The interviews had a duration of 20 – 30 minutes and have been carried out through face to face meetings. For the second component, data consists of observations made for the entire duration of the project. We had the chance to observe and participate in various daily routines of the team. On an ongoing basis we had discussions with team members regarding their tasks, respective role, responsibilities, and challenges in the software development process. For the third and fourth components, data collection consisted of online surveys utilizing a company internal tool to execute surveys targeting the engineering team. The first survey (S1), formulated based on the initial observations, aimed to obtain the overall developer satisfaction and the key challenges adopting DevOps practices in the project. To verify the relevance of the survey questions, S1 was sent out to a subset of the team and its intend and clarity were discussed with participants. While content reach, the data collected using S1 was not conclusive as the questions have been too generic and specific development requirements (e.g. UI focused development versus API focused development) have not been taken into consideration. Based on these observations the second survey (S2) was structured around system components (e.g. "How satisfied are you with the test automation framework when developing satellite services (C4)) providing additional component specific questions to explore the DevOps adoption specific to an area of the system. The surveys adopted Microsoft's Net Satisfaction (NSAT) (Microsoft 2013). The score ranges from 0 to 200. NSAT is calculated by asking a question in a survey using a 5-point Likert scale. The following options have been used: Very dissatisfied; Dissatisfied; Neither satisfied nor dissatisfied; Satisfied; Very satisfied. NSAT is then calculated by using the percentage of Very Satisfied responses, then subtracting the percentage of Very Dissatisfied and Dissatisfied responses, and then add 100.

Main preliminary findings

Utilizing the interview transcripts and the 204 answers to survey S2, we identified several factors that influence DevOps adoption for mission critical software. For the lowest NSAT scores of the survey, the detailed responses have been classified and categorized to identify the factors. The factors have then been interrelated with the interview data to analyze the relationship between the factor and potential remedies. The results are summarized below (table 1):

Factor	Observation	Remedies
Avoid compromising (F1)	S2 highlighted that developers are reluctant to change the system to avoid compromising the production environment.	Excellent test coverage and test frameworks to verify the change before it is rolled out to production. Flighting and feature switches to roll back quickly.

Factor	Observation	Remedies
Need for compliance (F2)	Mission critical software needs to fulfill certain audit and compliance requirements	Tools to support compliance tasks (e.g. Required code reviewer, Privacy data tagging)
Emergence of redundant tasks (F3)	We found that within each DevOps team similar tasks (e.g. Maintenance of CI/CD pipeline) are performed. These tasks require specialized knowledge that most of the developers do not have.	Introduce shared services organization that takes care of these tasks and automates them wherever possible.
Developer satisfaction (F4)	The surveys and interviews clearly highlighted that developers are not satisfied with the additional responsibilities and tasks prescribed by the new processes	Additional training and a feedback system (Post mortem reviews)

Table 1. Factors influencing DevOps adoption for mission critical software

Preliminary discussion and conclusion

In this case study, we analyzed factors affecting adoption of DevOps for the development of mission critical software. The contribution in this paper lies especially in lessons learned from the case study hence filling the identified research gap regarding a deeper understanding of DevOps and its adoption for mission critical software development. It is interesting to notice that, at a micro level, bimodal IT was adopted for the development of the monolithic kernel (C1). Thus, for the mission critical component of the product, traditional software development methods are still adopted. The satellite services however, fully adopted agile methods. By triangulating data from the interviews, surveys, and observations we noticed that this was just a temporary state. Teams developing C1 are slowly adopting DevOps practices, but are struggling as the toolsets and test coverage is insufficient for C1. Interestingly, bimodal IT appear a suitable, but temporary transition stage for the evolutionary development within a larger transformational process (Haffke et al. 2017). This case study provides preliminary evidence that DevOps improves the agility and reduces the time to market for the development of mission critical software. However, it poses challenges mainly concerning the fear of developers to compromise the system and the need for compliance. The first two factors can be overcome through technology. DevOps requires specific management of people to overcome some of the dissatisfaction within the teams. F3 needs to be researched further. At the current point it is unclear if the highlighted tasks are solely related to DevOps adoption for mission critical software or with the complexity and size of the software project. What we could observe within this case study was that to overcome some of the challenges a new DevOps unit has been formed, which provides shared services like build, monitoring and telemetry pipelines to the other DevOps teams.

This paper summarizes the preliminary findings of an ongoing research project aiming at investigating the factors influencing the adoption of DevOps practice for mission critical software. The adoptions and adaptations of DevOps for non-mission critical information systems are widely studied in prior literature. This paper describes research that focuses on DevOps within an area for which classical waterfall based methods are prescribed (Gartner 2015) and expands the existing literature. The identified factors need to be extended into concrete guidelines for organizations applying DevOps for mission critical software.

REFERENCES

Davis, J., and Daniels, R. 2016. *Effective DevOps: Building a Culture of Collaboration, Affinity, and Tooling at Scale*, (1st edition.), Beijing; Boston: O'Reilly Media.

Erich, F. M. A., Amrit, C., and Daneva, M. 2017. "A Qualitative Study of DevOps Usage in Practice," *Journal of Software: Evolution and Process* (29:6).

Fitzgerald, B., and Stol, K.-J. 2017. "Continuous Software Engineering: A Roadmap and Agenda," *Journal of Systems and Software* (123), pp. 176–189.

Gallivan, M. J. 2001. "Organizational Adoption and Assimilation of Complex Technological Innovations: Development and Application of a New Framework," *SIGMIS Database* (32:3), pp. 51–85.

Gartner. 2015. "Gartner IT Glossary." (https://www.gartner.com/it-glossary/, accessed January 19, 2018).

Ghantous, G. B., and Gill, A. 2017. "DevOps: Concepts, Practices, Tools, Benefits and Challenges," *PACIS 2017 Proceedings*.

Haffke, I., Kalgovas, B., and Benlian, A. 2017. "The Transformative Role of Bimodal IT in an Era of Digital Business," *Hawaii International Conference on System Sciences 2017 (HICSS-50)*.

Horlach, B., Drews, P., Schirmer, I., and Boehmann, T. 2017. *Increasing the Agility of IT Delivery: Five Types of Bimodal IT Organization*, presented at the Hawaii International Conference on System Sciences 2017 (HICSS-50).

Humble, J., and Farley, D. 2010. *Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation*, (1st edition.), Upper Saddle River, NJ: Addison Wesley.

Kranz, J., Leonhardt, D., and Kolbe, L. 2016. "The Enabling Effect of Collaborative Innovation in Information Technology Outsourcing on Individual Intrapreneurial Behavior," *ICIS 2016 Proceedings*.

Lwakatare, L. E., Kuvaja, P., and Oivo, M. 2015. "Dimensions of DevOps," in *Agile Processes in Software Engineering and Extreme Programming* (Vol. 212), C. Lassenius, T. Dingsøyr, and M. Paasivaara (eds.), Cham: Springer International Publishing, pp. 212–217.

Microsoft. 2013. *Determining Net User Satisfaction*, Microsoft. (http://download.microsoft.com/download/3/2/2/32269687-F181-449A-8C72-317090403C70/Determining_Net_User_Satisfaction.docx).

Mullaguru, S. N. 2015. "Changing Scenario of Testing Paradigms Using DevOps – A Comparative Study with Classical Models," *Global Journal of Computer Science and Technology*.

Poba-Nzaou, P., and Uwizeyemungu, S. 2013. "Barriers to Mission-Critical Open Source Software Adoption by Organizations: A Provider Perspective," *AMCIS 2013 Proceedings*.

Shropshire, J., and Sweeney, B. 2017. "On Devops and Workforce Morale," *SAIS 2017 Proceedings*.

Sommerville, I. 2015. *Software Engineering*, (10th edition.), Boston: Pearson.

Spurrier, G., and Topi, H. 2017. "When Is Agile Appropriate for Enterprise Software Development?," *European Conference on Information Systems 2017 (ECIS-25)*, pp. 2536–2545.

Wiedemann, A. 2017. *A New Form of Collaboration in IT Teams - Exploring the DevOps Phenomenon*, PACIS 2017 Proceedings.