On the Impact of Mixing Responsibilities Between Devs and Ops

Kristian Nybom(✉), Jens Smeds, and Ivan Porres

Faculty of Science and Engineering, Information Technologies,
Åbo Akademi University, Vesilinnantie 3, 20500 Turku, Finland
kristian.nybom@abo.fi

Abstract. Many software engineering organizations around the world are adopting DevOps. One of the goals of DevOps is to foster better collaboration between development and operations personnel, in order to improve organizational efficiency. Since DevOps is lacking a common definition, there are several approaches to adopt it, and organizations largely need to determine how to apply DevOps for themselves. In this paper, we present results from a case study in which a software organization adopts DevOps. The focus of this research is to study the impact of mixing the responsibilities between development and operations engineers. We interviewed 14 employees in the organization during the study, and results indicate several benefits of the chosen approach, such as improved collaboration and trust, and smoother work flow. This comes at the cost of a number of complications, such as new sources for friction among the employees, risk for holistically sub-optimal service configurations, and more.

Keywords: DevOps · Software process improvement · Adoption benefits and challenges

1 Introduction

DevOps has in recent years gained interest in the software and service development industry, and its adoption rate is expected to grow over the coming years [1]. DevOps addresses the challenge of what is often described as a gap between development and operations personnel. The gap is reduced through a combination of processes, cultural enhancements, and supporting technologies. More specifically, DevOps encompasses automation for reducing manual effort and improving stability, continuous feedback using metrics for improving software development processes, and a culture of collaboration and information sharing between teams [2]. However, the term "DevOps" is still an ambiguous concept and is lacking a standard definition [3–5]. While the purpose of DevOps is clear, organizations adopting DevOps must interpret and define what DevOps means to them.

The fact that DevOps is lacking a standard definition implies that there is no simple approach to follow when adopting DevOps in an organization. Adopting DevOps may thus not be a straightforward task since it may require that an

H. Sharp and T. Hall (Eds.): XP 2016, LNBIP 251, pp. 131–143, 2016.
DOI: 10.1007/978-3-319-33515-5_11

organization introduces process, personnel and technological changes and innovations. Since DevOps focuses on principles for software and service development, rather than specifying exactly how to implement DevOps, it means the path to a successful DevOps adoption is unique to each organization [6]. Therefore, we feel that it can be beneficial to learn from the successes and challenges experienced during previous DevOps adoptions when planning new DevOps initiatives.

In this article, we present how a particular organization adopted DevOps and what the impact was of this adoption from the perspective of engineers. In particular, we study the impact of mixing responsibilities between development and operations personnel, and how this affects the culture, tools and work practices. The study was carried out by interviewing both development ("Devs") and operations ("Ops") personnel before the start of the DevOps adoption and six months into the adoption.

2 Background: Approaches to DevOps Adoption

DevOps is commonly viewed as a professional movement that emphasizes communication, collaboration and integration between software developers and IT operations, see e.g. [7]. According to Willis [8], DevOps is comprised of four key aspects: *culture*, *automation*, *measurement* and *sharing*. In our previous work [9], we described DevOps as a number of *engineering process capabilities* supported by certain cultural and technological *enablers*. According to this definition, the capabilities define processes that an organization should be able to carry out, while the enablers support efficient work execution of these processes.

Common to most definitions of DevOps is that one of the main goals behind it is to tackle the problem of having development and operations teams in functional silos (see e.g. [10]) – a problem which is often present in non-DevOps software development organizations. The teams are in functional silos when there is little support for communication and collaboration between them in order to make releases. Breaking down the silos improves the development cycle, by bringing Devs and Ops closer to each other, allowing the organization to produce more production-ready code and deliver better services more frequently. Breaking down existing silos is, however, a non-trivial task, and it is tightly coupled with improvements in work processes, culture, and technology.

Focusing on concrete actions that address the problem of bringing Devs and Ops closer to each other, three possible but distinct approaches are to

1. Mix responsibilities: assign both development and operations responsibilities to all engineers, or
2. Mix personnel: increase communication and collaboration between Dev and Ops, but keep existing roles differentiated, or
3. Bridge team: create a separate DevOps team that functions as a bridge between Devs and Ops

Which approach to use may be difficult to decide on. An argument for following Approach 1 (mix responsibilities) lies in the concept of *Infrastructure*

as Code. What this concept refers to is that the infrastructure for deploying software is fully automated, and is controlled by code. As mentioned in [11]:

> "If infrastructure is code, then almost by definition, infrastructure becomes to some degree a function of development, or at least so hard to separate from development that the distinction becomes almost irrelevant."

Assuming that infrastructure is code, this statement suggests that Approach 1 (mix responsibilities) is a natural approach, because Ops will be involved in Dev tasks by developing the infrastructure together with the Devs.

As for Approach 2 (mix personnel), it is stated in [12] that creating cross-functional teams is a good approach when adopting DevOps. These teams should consist of Devs, testers, Ops personnel and others, and then each of them would contribute code to a shared repository. In this way, the Dev and Ops responsibilities are maintained, but communication and collaboration is promoted. It is also mentioned in [12] that although promoting communication and collaboration is key, training for Devs and Ops on the responsibilities of other departments can be very beneficial for communication.

In a blog post [10], Jez Humble strongly states that Approach 3 (bridge team) should not be followed when adopting DevOps, since a separate DevOps team will not break any silos, but instead create new ones. Nevertheless, [13] reports that DevOps departments are a growing trend, and that according to their survey, more than 90 percent of those working in DevOps departments are in companies with medium to high IT performance.

3 Research Questions and Study Design

We consider that there is a need for empirical studies describing how DevOps is being adopted in different organizations and for the benefits and drawbacks of adopting DevOps. In this article, we decided to focus on the DevOps approach based on mixing responsibilities, and left studies of other approaches for future work. The main research question is as follows

RQ What may happen when mixing responsibilities between developers and operations teams in an existing organization?
RQ.a How does this approach affect the culture?
RQ.b How does this approach affect the tooling?
RQ.c How does this approach affect the ways of working?

This research was done as a longitudinal case study: we observed an organization as the phenomenon happened. For collecting data for the study, we used semi-structured interviews of company employees. For selecting the organization for the case study, we had the following two criteria:

1. Before the start of the DevOps adoption, there has to be clearly separated roles between Devs and Ops in the organization
2. The organization chooses Approach 1 as part of their DevOps adoption

The selected organization was an international IT company with a long history and over 1000 employees, which develops both software and services for customers. The case organization contains several organizational units, each having their own R&D teams. These units are combined by a separate operations unit.

This study was carried out in one of the organizational units, which develops and operates in the cloud services area. That unit was also the only unit in the organization that was actively adopting DevOps. Motivations for the adoption were to make software deployments faster and more frequent, to share knowledge between development and operations, and to keep deployment costs low.

A total of 14 experienced employees were selected by the company so they would represent different work areas, e.g. development, quality assurance, operations, and management. Their familiarity with DevOps prior to the study varied from understanding the basics of the concept to having previous professional experience of successfully adopting DevOps.

We conducted two rounds of interviews. The first interviews were conducted in the end of May 2014. Before the interviews, the participants were informed about the study, that the interviews will be recorded and that the answers will be handled anonymously. The interviews lasted roughly 45 min on average. An interview guide containing a broad field of questions was used for the semi-structured interviews. The purpose of the first round of interviews was to get an overview of the organization, of their processes, of the daily work, and of employees' expectations and concerns regarding their DevOps adoption. The recordings of the interviews were transcribed, coded and analyzed, and some results from that round have been reported in [9].

The second round of interviews were conducted in October 2014, and followed the same procedure as the first round. The questions for the second round were designed based on the results from the first round, and many of them were angled to expose changes since the first round. Other topics covered were related to the software development processes, relationship between development and operations, teamwork, employees' feelings (such as pressure, impact and importance regarding his/her work), how the DevOps adoption had proceeded along with expectations and concerns, views on the management, and the DevOps aspects of automation and familiarity with others' work.

The recordings from the second round were transcribed. Thereafter they were coded separately by two of the researchers to make sure that no relevant information would be missed, and to reduce the risk of researcher subjectivity influencing the codes. The researchers used slightly different approaches to code the material. The first coding approach was as follows. First, the transcripts were read though and summarized to obtain a quick overview of the subjects discussed in the interviews. Then, the transcripts were read through in detail with the researcher identifying, assigning, comparing and adjusting codes according to the content. Finally, the transcripts and the corresponding coding were read through once more from the start to check and make some final adjustments to the codes.

A second coding approach was to use pre-defined codes according to the research questions in this article. While reading through the transcripts and

assigning content to the codes, different subjects discussed during the interviews were simultaneously identified. A second round of coding was then done within each of the pre-defined codes, using the identified subjects as pre-defined sub-codes. This resulted in detailed codes for each research question. The researchers then individually identified what was perceived as beneficial or challenging from both coding approaches. The individual lists were then compared, discussed and merged into our final list of outcomes. The results presented in this article are based on the second round of interviews.

4 Results

Before presenting the outcomes of the interviews, it is worth mentioning that the organizational structure, or more specifically, the fact that operations were attending the products for all the different organizational units, had an impact on several of the things mentioned below. Most notably, this resulted in operations having limited possibilities in taking on development responsibilities. Another fact to notice is that only one of the organizational units were actively adopting DevOps, while the other units were not, resulting in a difficult situation for operations: depending on which unit they were attending, they needed to work according to a specific pattern.

In the following we use the terms "Dev" and "Ops" to describe engineers with previous experience and responsibilities within software and service development and operations respectively.

4.1 Impact on Culture

A New Source for Friction. In order to enable Devs to deal with operations tasks, it was necessary to give administration rights to Devs to different environments. Based on the comments from the employees, it was evident that gaining access served as a cause for friction and mistrust. It was also mentioned that the process for obtaining access was long and tedious.

The long process also had negative implications on the work efficiency, because employees often realized too late that they needed the access, causing extra delays. The decision process for who was granted the access was also described as unfair. Some employees mention that access seem to have been granted based on shown interest rather on experience and knowledge. Devs also complained about Ops getting access faster than Devs. This made them angry and irritated.

An Eye-Opening Experience. Mixing the responsibilities of Devs and Ops was considered educating for the Devs. In the organization, the Devs had been developing various tools for their operations personnel to use for a long time, but only now with the DevOps adoption initiative did they get to see how their own tools were working.

Seeing the operations side also surprised the Devs in the sense that they now realized how far from production ready their software usually was, although it had passed all the tests in their own environment.

Through teaching others and learning from others, Devs and Ops were beginning to trust each other more. The increased level of trust was accompanied with stress relief, specifically for operations personnel as they could trust Devs to do part of the operations tasks. As a consequence, knowledge about operations tasks and problems were increased among the Devs. This lead to Devs starting to improve test environments to better correspond with production environments, while also contributing to increased collaboration between Devs and Ops.

Learning how to do operations tasks was not straightforward for everyone. Some employees mention it being extremely challenging, and that they did not see the point in having Devs do tasks which other more proficient employees do better. The complications in learning how to do operations tasks resulted in a certain reluctance in learning new things among the Devs. These Devs mentioned that they would prefer having the distinction between Devs and Ops more clear, implying that the mixing of responsibilities were not to their liking. Additionally, learning how to do upgrades was considered time consuming, but on the positive side, it had also revealed flaws in the upgrade processes. Devs mentioned the greater need for knowledge and expertise, since they now were responsible for everything and consequently needed to know every technology used. This was visible as mixed feelings among the Devs.

Shared Responsibilities. The view on how responsibilities were shared varied. Devs largely felt that responsibilities were shared, and if something went wrong, it was everybody's fault, while some Ops felt that Devs were somewhat unaccountable, specifically when it came to fixing problems late in the evenings. Their opinion was that Devs wanted to decide on everything, how the product is designed, how it is deployed, etc., without involving operations personnel. Then at the end of regular office hours, Devs would not care anymore and would want Ops to take care of it.

Employees agreed that within development, the responsibility of deploying software was shared among the Devs. They mentioned that whenever someone had problems with deploying software, they simply needed to shout it out, and everyone was alert and helping that person if needed.

Improved Collaboration. Mixing the responsibilities brought Devs and Ops closer to each other. Employees mentioned that Devs and Ops now collaborate on different tasks, since they now realize the importance of collaboration. Everyone agreed that collaboration between Devs and Ops is good on an individual level, and to some extent also on team level, but some employees called on the support from managers to further improve collaboration by providing more reasons for collaboration. It was mentioned that through the improved collaboration, it was easier to get things moving forward, since Devs could discuss directly their issues with Ops personnel, which is much faster than having to contact managers to get the issues solved.

On the other hand, Ops felt uneasy about Devs coming into their domain, and mentioned that adjusting to this takes time. Additionally, the closer collaboration and specifically keeping Devs and Ops synchronized was described being time consuming. It was argued that, although individual, the work space affects the level of collaboration to some degree, since long walking distances might imply a threshold for going to talk to some other person.

Through the collaboration, both Devs and Ops had become more trusting and understanding towards the other. Ops had seen that Devs can do the operations tasks without jeopardizing service stability and Devs had realized what Ops have to struggle with in order to deploy their software.

4.2 Impact on Internal Development Tools

Awareness of Tool Quality. As mentioned earlier, Devs had been developing tools for their operations personnel, and now that Devs were dealing with operations tasks, they were using their own tools. Devs mentioned that they were now experiencing the flaws and problems that the tools had, something which Ops had been aware of all the time. But now that Devs were using their own tools, and since they were not accustomed to having poor solutions, they were putting extra effort into creating very good tools for deployment. Development of these improved tools was performed in collaboration with Ops.

Deployment Risks. Previously operations was the place where the entire service stack came together, where all problems materialized, and where decisions were made which affected the entire service stack. Since Devs had been given the power to deploy their own product, there was some concern that they could make decisions that would be optimal for their specific product, while unknowingly disregarding the impact of their decisions on the remaining service stack. The main risk identified was that problems caused by these kinds of decisions are realized too late.

Identified Tooling Obstacles. It was mentioned that the many environments and many ways of upgrading different services creates an obstacle for full automation. Ops mentioned that automatic reactions to various glitches that may occur cannot be defined. Ops always have to investigate those problems manually. These problems were partially realized by Devs too. They perceived deployment as being time consuming and requiring significant effort, and while they technically could create scripts that would deploy everything, the real problem was to create scripts that recover from glitches. Another concern mentioned was that without automation, configuring all the different environments correctly is error-prone, specifically when there is a change in configuration. The Devs felt that it is easy to forget to align the configurations across all the different environments.

4.3 Impact on Ways of Working

Added Responsibilities. According to the chosen DevOps adoption approach, Devs were now responsible for performing upgrades on certain production environments. These environments were pre-staging environments, in the sense that they were mostly for internal users. A so called build master role was introduced among the Devs, which would rotate within the team on a weekly basis. In addition to doing the upgrades, the build master was also required to debug and investigate the production environment.

Devs mentioned that getting used to the build master role, and focusing on it was demanding – it is easy for Devs to start working on something else as soon as they have completed their task as build master, even though they noticed something that should be fixed.

Benefits of Having Administration Rights. The perceived benefits of Devs having access to different environments were manifold. Getting e.g. statistics from the production environments was described being considerably easier through the granted access, making work much smoother. Devs mentioned that it also allows better debugging, because Devs do not need to ask Ops for help anymore. It is faster, and more thorough, because Ops do not always have time to delve into the problems. On the other hand, this is time away from feature development. Ops also said that they had many times received help from Devs in problematic situations, making their work easier.

Common Ways of Working. Employees mentioned that the mixing of responsibilities puts higher requirements on common work practices and technical solutions between Devs and Ops. Without this, a risk identified was that Devs create tools specific for only their own unit's needs rather than having common solutions for all the organization. They mentioned that the upper management needs to push for common solutions in order to avoid this situation. They also mentioned that without strong management, the increased freedom among the Devs may result in a chaotic working environment, where everyone is doing as he or she pleases.

A concern was that even though access had been granted to Devs and employees had new responsibilities, work was done quite far in the same way as earlier. Other concerns among the Ops were that with added responsibilities and granted access for the Devs, Ops responsibilities had changed towards support, and that Devs were making decisions without consulting Ops.

Devs occasionally dealing with operations tasks was mentioned to have negative implications on the employees' work flow, as they caused complicated context switches. They said that it is easy to switch between tasks when they are within the same area, but switching between development and operations tasks is complicated. These context switches were perceived as frustrating.

Concerns with Mixed Responsibilities. Several concerns in the chosen approach of adopting DevOps were also discussed. It was mentioned that people like

to do what they are used to doing. Thus, introducing operations tasks to Devs was perceived as complicated, and Devs would try to avoid them. Devs were used to making their own engineering decisions, but this was described as problematic, since they now created their own solutions also for operations tasks, instead of learning from others and reusing common solutions.

The combined effort of Devs and Ops was described having its own complications, because more people making changes to configuration and software leads to an increase in the probability of error, simply because the tools and processes for performing such changes have not emerged. To cope with this situation, employees felt that there is a great need for guidance, and will for involvement from everyone, so that they can agree on a common approach.

Devs doing operation tasks was experienced both positively and negatively. Some people loved having control over the entire delivery chain, while others wondered why not more experienced people could take care of the deployment. Several employees felt, however, that the chosen approach of mixing responsibilities was the wrong approach for doing DevOps. A perceived problem was that technical systems were tied to specific APIs and then Devs and different development teams were given too much freedom in choosing their own way of doing things. With many such development teams, a risk mentioned was that the organization ends up with many ways of working, causing lack of synergy.

5 Discussion

The results from the interviews indicate several beneficial aspects when mixing responsibilities between Devs and Ops. Devs have seen what work is required in order to deploy their software, which is educating for them. In addition to allowing them to develop more production ready code, it also reveals problems and flaws in some of the tools they have developed for the Ops. As a consequence, Devs are now putting more effort into developing better tools, which is done in collaboration with Ops. This clearly shows a benefit of learning about responsibilities of other teams. When Devs learn what happens with their code after it is developed and tested, they can exploit this knowledge for producing better code in the future. Unfortunately, corresponding benefits for the operations personnel were not revealed, because they were unable to take on development responsibilities. This was mainly due to the organizational structure, which required the operations teams to deal with software from all the organizational units.

Both the collaboration and trust between Devs and Ops is improved through the mixed responsibilities. Instead of contacting managers to solve problems, employees can discuss directly with personnel from the other team which is much faster. Ops have realized that Devs can deal with the operations tasks they are assigned with, without jeopardizing the stability of the service. Devs, on the other hand, have seen what Ops have to go through in order to deploy their software. This weakens the silo structure between the Devs and Ops, and the teams are effectively collaborating more. The weakened silos also inspire employees for even more collaboration, and some employees said that they would want the

managers to give them even more reasons for collaboration. Thus, mixing the responsibilities seems to weaken the silos, as Devs and Ops are encouraged, and even required, to communicate and collaborate more.

Giving administration rights to Devs was seen as beneficial in many ways. Devs get statistics from production environments making work smoother, they can fix errors more easily than previously, and fixing errors is more thorough and efficient. The drawback is that all of this is time away from feature development.

The chosen adoption approach was not without complications. Surprisingly, getting administration rights was described as a source for friction, since employees felt that administration rights were not granted on a fair basis. Dealing with operations tasks was far from an easy task for several Devs, and because of this, the opinion of having separate responsibilities was strengthened among them. The organizational structure prevented Ops to fully take part in the DevOps adoption, since they already had their hands tied with operations tasks for other organizational units. We believe that this fact also partially prevented the teams from developing common ways of working, since Ops also had to work with other units that were not adopting DevOps. The concerns associated with taking on operations tasks among the Devs are a natural reaction. It is understandable that they wonder why they have to deal with operations tasks when there already is more proficient personnel to deal with those tasks. In general, a certain reluctance towards the adoption of DevOps was observed.

Devs having the power to deploy their own software was repeatedly mentioned as dangerous because this could potentially damage the entire service stack in the organization. The reason is that Devs were not aware of software developed in other organizational units, and consequently were configuring their software without those in mind. This presents a risk with the adoption approach, because if other organizational units had had similar power, it could have produced a chaotic end result, where all units would create their own solutions. With a lack of collaboration, communication and shared work practices and goals between Devs and Ops, this risk is further strengthened. To improve the situation, management could actively try to improve inter-team relations in order to facilitate communication to ensure that information spreads across teams. Automation could also assist in solving this problem to some degree.

Creating fully automated deployment tools was mentioned being a necessity for a well-functioning DevOps implementation. With the many different environments and many ways of upgrading, employees were of the opinion that automatic reactions to various glitches that may occur cannot be defined. Consequently, a large effort was continuously put into configuring and upgrading software, and employees called for a holistically well-functioning deployment tool chain. The effort required to put into this also had other implications, since it required Devs to make complicated context switches between development and operations tasks. Better automation could have assisted with the context switches, improving the work flow of the employees.

When both Devs and Ops independently make changes to configuration and software, there is a greater probability of error, as long as the tools and processes for performing such changes are not improved. This clearly shows the need for developing common ways of working and improving the automation.

It is clear from the respondents that what DevOps means to the organization should be clearly communicated to the employees in order to support a successful adoption. Currently, Devs felt that DevOps mostly meant that they get additional operations tasks to deal with once in a while, and when they are completed, they go back to developing. Clearly, this view is counterproductive for improving collaboration between the teams, and to avoid this, guidance and instructions are a necessity.

Had a third round of interviews been performed later into the adoption process, it is likely that the collaboration between the teams had been further improved, and that DevOps had stabilized more. When the second round of interviews was performed, however, employees were still adapting to the new responsibilities. Since people change slowly, it is not surprising to see certain instability, uncertainty, and reluctance among the employees.

6 Conclusions

This paper describes phenomena that arose when mixing responsibilities between developers and operations personnel in an organization when adopting DevOps. The results are from a case study, in which a software organization adopting DevOps was studied. The case organization consisted of several organizational units and a separate operations unit. In the organization only one organizational unit was adopting DevOps, which impacted the results in the sense that operations were not fully able to participate in the adoption.

The results indicate several benefits of the mixed responsibilities. Collaboration and trust were improved between Devs and Ops, and seeing what the other team has to deal with was very educating, helping employees in their work. Through increased collaboration, the work flow was described as smoother and faster as compared to earlier. Since Devs were dealing with operations tasks, they realized problems and flaws in the tools that they had earlier developed for operations, and Devs were now working on improving the tooling.

Several complications with the chosen adoption approach were also revealed. As Devs had the power to configure and deploy their own software, a major concern was that they would create solutions that were optimal for their software, while unknowingly disregarding the impact this had on the remaining service stack. The lack of common ways of working between Devs and Ops reinforced this concern. Dealing with new responsibilities among the Devs was considered challenging by many, and even strengthened their opinion of having separate responsibilities. Because of the challenging operations tasks, Devs realized the importance of having automated infrastructure, but accomplishing this was described as being extremely complicated in the case organization.

Finally, the study reveals the need for a strong management when adopting DevOps, since Devs and Ops need to develop common goals, practices of deploying, and approaches to technical solutions. The management also needs to clearly communicate to the employees what DevOps means to the organization, so that the personnel will realize the reason for the adoption, and the requirements and

benefits of it. Automation of the infrastructure is of key importance, specifically when Devs are given the responsibility of configuring and deploying their own software.

The results indicate that when operations work with several organizational units, it is challenging to adopt DevOps in only some of those units. Thus, the overall organizational structure may impact the DevOps adoption process.

Acknowledgements. This work has been partially supported by the Digile Need for Speed program and funded by Tekes, the Finnish Funding Agency for Technology and Innovation.

References

1. Gartner, Inc.: Gartner says by 2016, devops will evolve from a niche to a mainstream strategy employed by 25 percent of global 2000 organizations. http://www.gartner.com/newsroom/id/2999017. Accessed 23 February 2016
2. Babar, Z., Lapouchnian, A., Yu, E.: Modeling DevOps deployment choices using process architecture design dimensions. In: Ralyté, J. (ed.) PoEM 2015. LNBIP, vol. 235, pp. 322–337. Springer, Heidelberg (2015). doi:10.1007/978-3-319-25897-3_21
3. Hüttermann, M.: DevOps for Developers, 1st edn. Apress, Berkely (2012)
4. Roche, J.: Adopting devops practices in quality assurance. Commun. ACM **56**(11), 38–43 (2013)
5. Capgemini, Devops - the future of application lifecycle automation, December 2014. https://www.capgemini.com/resources/devops-the-future-of-application-lifecycle-automation. Accessed 17 December 2015
6. Virmani, M.: Understanding devops & bridging the gap from continuous integration to continuous delivery. In: 2015 Fifth International Conference on Innovative Computing Technology (INTECH), pp. 78–82, May 2015
7. New Relic, What is devops'? http://newrelic.com/devops/what-is-devops. Accessed 18th December 2015
8. Willis, J.: What devops means to me, July 2010. http://www.getchef.com/blog/2010/07/16/what-devops-means-to-me/. Accessed 3 December 2014
9. Smeds, J., Nybom, K., Porres, I.: DevOps: a definition and perceived adoption impediments. In: Lassenius, C., Dingsøyr, T., Paasivaara, M. (eds.) XP 2015. LNBIP, vol. 212, pp. 166–177. Springer, Heidelberg (2015)

10. Humble, J.: There is no such thing as a "devops team", October 2012. http://continuousdelivery.com/2012/10/theres-no-such-thing-as-a-devops-team/. Accessed 17 December 2015
11. Riley, C.: Do, should developers own infrastructure? June 2015. http://devops.com/2015/06/25/doshould-developers-infrastructure/. Accessed 17 December 2015
12. Wade, E.: In devops culture, communication, collaboration are key. https://www.veracode.com/blog/2015/07/devops-culture-communication-and-collaboration-are-key. Accessed 27 December 2015
13. Puppet Labs, New Relic and Thoughtworks, "2014 state of devops report" (2014). http://puppetlabs.com/sites/default/files/2014-state-of-devops-report.pdf. Accessed 23 February 2016