DOSSIER-Cloud

DEVOPS-BASED SOFTWARE ENGINEERING FOR THE CLOUD

http://www.dossier-cloud.eu

Deliverable D2.3

Survey Paper / Technical Report on DevOps-Oriented Software Engineering

Document details:

Editor :	Andreas Christoforou
Contributors :	Andreas Andreou, Adonis Podinas, Constantinos Stylianou, Mike Papazoglou, Willem-Jan van den Heuvel, Luciano Baresi, Clement Quinton, Srdjan Krstic
Date:	21 June 2017
Version:	8.0

Document history:

Version	Date	Contributor	Comments
1.0	05/06/17	Andreas Christoforou	Initial document, structure and content
2.0	06/06/17	Adonis Podinas	First draft
3.0	15/06/17	Andreas Christoforou	Corrections, second draft
4.0	17/06/17	Andreas Andreou, Luciano Baresi, Clement Quinton, Srdjan Krstic	Corrections, third draft
5.0	18/06/17	Mike Papazoglou, Willem-Jan van den Heuvel, Andreas Christoforou	Corrections, fourth draft
6.0	19/06/17	Constantinos Stylianou,	Corrections, fifth draft
7.0	20/06/17	Andreas Andreou	Corrections. Final review for approval
8.0	21/06/17	Luciano Baresi, Mike Papazoglou	Approved final version after minor revision

Contents

1. Introduction

Targeting faster application delivery and born from agile methodologies, the DevOps approach has recently started to be applied on software development and operation processes. Among various process tasks like testing, deployment, reconfiguration, etc., the DevOps approach favors a new culture between development and operations teams aiming to achieve a closer collaboration and communication without silos and barriers between these teams, and to emphasize automation and sharing. DevOps describes the main stakeholders of producing and supporting a distributed software service or application that possess a mixture of code skills and system operation skills (Duvall, 2012).

Humble and Molesky (2011) state that DevOps consists of four dimensions: culture, automation, measurement and sharing. Culture mainly refers to organizational structure and the corresponding new processes to support collaboration. Also, culture takes into consideration the efficient management of human resources in activities such as team staffing. Automation of practices is the goal behind the support of new adopted tools and processes. Measurement in DevOps is defined by the authors as "monitoring high-level business metrics such as revenue or end-to-end transactions per unit time". Metrics at a lower level are important for measuring the delivery process, as well as the way people work. Sharing is an essential component towards knowledge and control spreading across teams.

The study of the DevOps phenomenon and its various aspects is closely related to cloud computing. Although cloud computing and DevOps are two independent computing/software paradigms or strategies where one does not prerequisite the other, very often they are used together under the modern software development process, which advocates in favor of delivering everything as a service.

Academic research on DevOps that also considers the industrial interest for adoption (investment) is very limited. Academic literature is mainly focused on the description and discussion of the DevOps approach, as well as the ways that it is used to support or to be supported by other subjects. In contrast to the academic literature, a quite large number of other resources, either technical reports or white papers, are available. In addition, one can easily discover that there is a lack of a concrete definition of DevOps and there are many perspectives with which it is approached; thus, DevOps can be defined as a set of best practices rather than a specific common framework.

This technical report follows a survey paper approach aiming to present a literature overview of the most significant issues on DevOps approaches reported in research literature, and how this recently emerging approach is affecting the area of software engineering. In the next section the methodology followed for discovering and including/excluding papers is presented, while section 3 introduces the findings and results of this study. Finally, section 4 concludes the report providing a summary of the findings and future work.

2. Methodology

This research study is motivated by two research questions:

RQ1: What is the scientific research devoted to the DevOps approach?

RQ2: Which are the main aspects of recent studies on the DevOps approach?

In order to answer these two research questions, we applied a methodology to gather enough material to justify the positions and statements made; this methodology is described in this section.

For the first step, we followed the guidelines for a systematic literature review proposed by Kitchenham (2007). Although a SLR is outside the scope of this work, this helped us to organize the process of finding and classifying relevant works. We searched for articles indexed in Scopus, Science Direct, IEEE Xplore, ACM Digital Library, SpringerLink, Google Scholar and Wiley Online. The search strings used were “DevOps” and “Developers and Operations”, “Development and Operations”. The search results consisted of articles published up to and including 2016. As DevOps is a very recent topic, we considered both journal and conference articles. Finally, we removed duplicate papers from the results, since the search engines and databases produced overlapping results to a certain extent. After these steps, the initial collection consisted of 92 potentially relevant works. Then, we performed a detailed, qualitative analysis by reading these papers in order to identify and suppress both different papers of the same authors/group incrementally reporting their results, and works that used the term “DevOps" with a different meaning. From this analysis we used a filtering process based on a set of four different types of criteria on the initial list of papers. This set of criteria consisted of the type of the paper, publication year, publication venue and number of citations. We accepted only scientific papers published in recognized venues with a significant number of citations. The final set of papers was organized into several categories and these are presented in the section followed.

3. Findings

In this section we present the findings and results aiming to adequately respond to the research questions that motivated this study. Also, some quantitative features of the study are depicted in the form of bar charts.

The final list of papers consisted of 47 papers which were organized in several categories based on the mining process, as well as their content. Figure 1 shows the final list of papers by year of publication. It is worth noting that in 2011 and 2012 only one paper was published each year. In the next year (2013) this number is slightly increased to 6 papers, while from 2014 to 2016 a

significant increase is observed with 12 papers in 2014, 16 papers in 2015 and 11 in 2016. This can be easily attributed to the fact that DevOps, as a relatively new concept emerging in the scientific community, did not attract researchers from the beginning, but gradually gained interest over time.

Figure 2 shows the categorization of the material by type. Five types of research papers were identified: chapters, conferences, journals, technical reports/white papers and MSc Thesis.

In their majority, the documents are either conference papers or journal articles. More specifically, we have 23 conference papers, 16 journal articles, six technical reports, one book chapter and one MSc thesis.

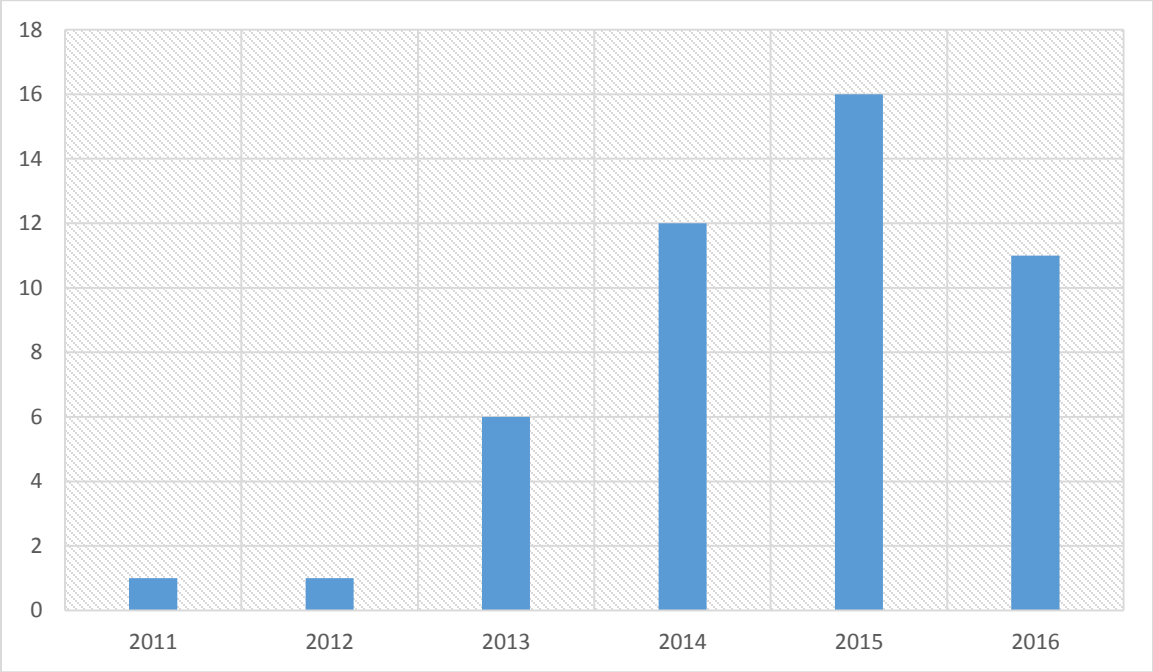

Figure 1: Number of papers examined by publication year.

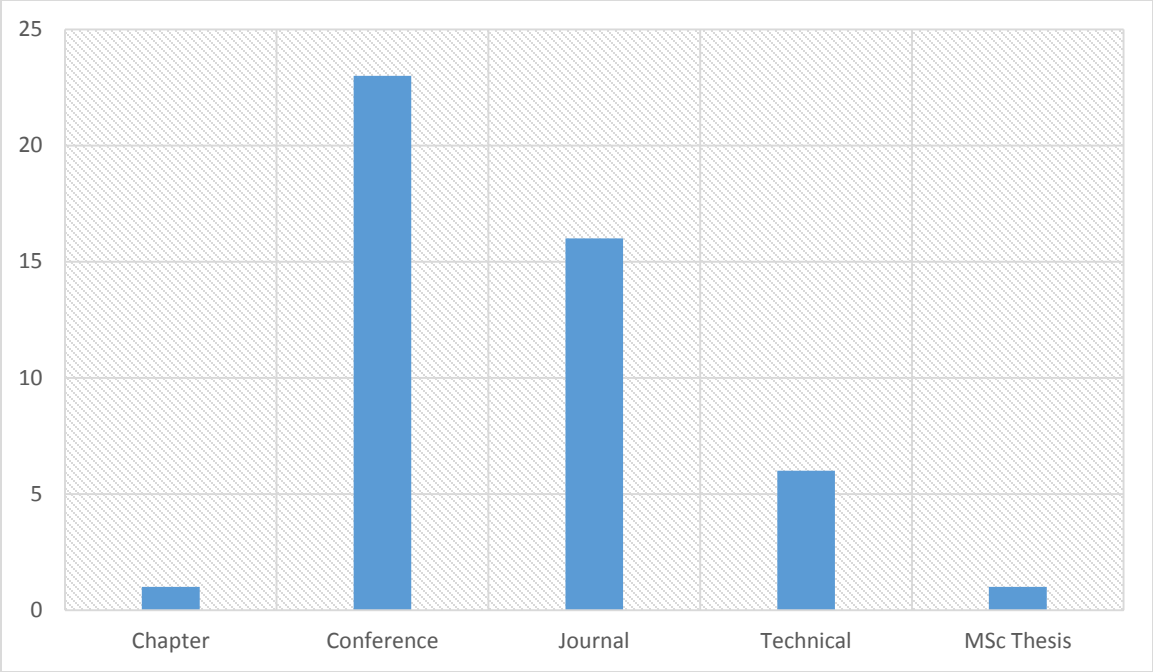

Figure 2: Number of papers examined by publication type.

Based on the content orientation, we split the papers into five categories: *definition*, *tools and methodologies*, *DevOps application, literature review* and *research challenges*. Papers belonging to the category *definition* mainly present a short history of DevOps and its various aspects and definitions, while those in the *tools and methodologies* category mainly focus on various tools, frameworks or guides which provide support to DevOps application. Accordingly, papers in the *DevOps application* category make reference to various examples where DevOps principles and methodologies have been applied. Review papers that aim to make a discussion about DevOps and to identify DevOps' definition and scope were categorized in the *literature reviews* category. Finally, the category *research challenges* consists of papers that refer to a series of research challenges.

Figure 3 shows the quantitative categorization based on the content orientation.

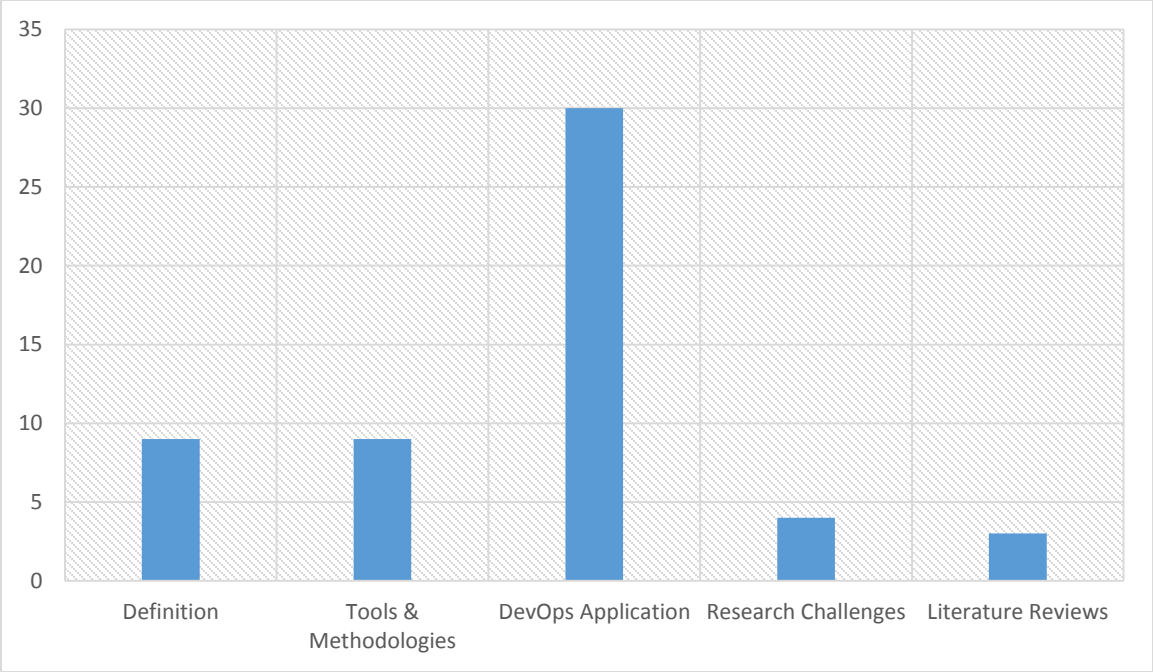

Figure 3: Number of papers examined per content orientation.

3.1 Definition

Akshaya et al. (2015) attempt to make an introduction to DevOps methodologies by giving a brief description of the four categories that DevOps is divided into, that is, log monitoring, build and test, deployment and configuration.

Dyck et al. (2015) identify a definition problem for DevOps and release engineering. The fact that there is no uniform or globally accepted definition for release engineering and DevOps has led to many inadequate or even wrong definitions to exist. Consequently, these terms are often confused or in other cases are used as synonyms. This paper, described those two terms by contrasting available definitions and descriptions for both of them. Additionally, uniform definitions for release engineering and DevOps were proposed, which were developed in cooperation with experts. The authors received consistent positive feedback for the definition of release engineering, while the definition for DevOps still needs discussion.

Liming et al. (2016) discuss the DevOps definition and how its practices affect developers throughout the software development lifecycle. The article also addresses microservices and gives a brief description of what their target is. Also, it defines the domains that might benefit from DevOps and argues about the kind of practices that are best to adopt according to the type of systems and organizations.

A brief overview of the most recent DevOps technologies is presented by Ebert et al. (2016). A particular reference is made to delivery tools and microservices and their meaning for industry projects is discussed. This study ends noting that because products and lifecycle processes vary, each company needs its own approach to achieve DevOps, from architecture to tools and culture.

The work of Lwakatare et al. (2016) is focused on consolidating the understanding of DevOps and its practices as described by practitioners using multivocal literature and interviews. The study contributes a scientific definition of DevOps and patterns of DevOps practices to help identify and adopt it. Evidence from how practitioners referred to the DevOps phenomenon also supports that DevOps is a mindset as stated in the proposed definition. However, the results showed that, in addition to being a mindset, DevOps constitutes a set of practices attributed to it. The authors argue that both mindset and practices must be included in the definition, and that the first part of the definition should not exclude the results. As a consequence, this paper improves the definition of DevOps being a mindset substantiated with a set of practices to encourage cross-functional collaboration between teams, development and IT operations within a software development organization, in order to operate resilient systems and accelerate delivery of change.

The technical report by Edwards (2010) discusses why DevOps is needed in the enterprise world and what is the relationship between cloud computing and DevOps. Also, it analyzes if the enterprise is ready for DevOps. There is also a short discussion about the infrastructure operation roles.

The goal of Velasquez et al. (2014) is to provide a picture of how DevOps works today, based on real-world data from real people. Eventually this picture aims at helping IT managers and practitioners understand how to build greater value in their teams and help their companies win in the marketplace arena. The authors introduce questions so as to gain a better understanding of the impact that IT performance and DevOps practices may have on the overall organizational performance by using metrics that matter to the business: profitability, market share and productivity. The analysis of the survey data showed that strong IT performance is a competitive advantage, DevOps practices improve IT performance, organizational culture matters and job satisfaction is the primary predictor of organizational performance. A relevant technical report is provided by Walls (2013), which gives a brief description of what organizational culture is and when a culture is identified as DevOps.

Peuraniemi (2014) aims to clarify the definition of DevOps and review its principles, methodologies and tools. The paper also deals with the characteristics of these areas, which give value to the customer and how these characteristics affect delivery. Also, the author aims to specify to which of the three areas of services customer satisfaction DevOps principles and methodologies apply. The definition of DevOps is given with contrast to other definitions. Additionally, the paper analyzes the DevOps principles and defines which of them can be seen as value-driven principles. Moreover, methodologies and tools are summarized and finally the paper refers to a case study on applying DevOps principles in a large company, such as IBM.

3.2 Tools and Methodologies

Ahmadighohandizi et al. (2015) state that when applications and services are deployed into the cloud, several factors need to be considered. Enough resources need to be allocated for the hosting of the application so as to ensure the required service level under the assumed usage load. Various legal and trust-related concerns can also affect the decision of the physical and virtual location of the service. Finally, the cost of running the service needs to be minimized. This research combined automation and brokered cloud to establish an integrated development tool. By using this tool, one can develop applications in a cloud-based IDE, find the most suitable cloud provider, and deploy the application in the target platform. To enable the fast development and delivery cycle, all the relevant steps are automated. Firstly, the cloud brokerage automatically finds the deployment target out of several cloud platforms. Secondly, automatic provisioning and deployment are performed through a unified interface for heterogeneous Platforms as a Service (PaaS). Finally, continuous deployment pipeline automates code integrations, builds, and deployments.

The key to DevOps success is the automation of software quality assurance. The work of Waller et al. (2015) presents how automated performance benchmarks may be included into continuous integration. As an example, the paper reports on regression benchmarks for application monitoring frameworks and illustrates the inclusion of automated benchmarks into continuous integration setups. Applying these automated benchmarks in an early stage of the development process enables an early detection and repair of performance issues before such issues are propagated into a release. This approach contributes to the current efforts of the DevOps movement by presenting a case study of executing and analyzing regression benchmarks in continuous integration setups.

Liu et al. (2014) examined the use of a software tool to accelerate the responses to customer change requests within a product development organization, namely IBM. It analyzes a comprehensive case study (from conception to deployment) of a prototype created by IBM's Rational Software AM release team to collect, analyze and monitor the flow of customer change requests for a set of products. The results of this work have led to a more accessible, frequent and informative tracking and reporting system that has significantly reduced the response delivery time for customers. This study also suggested a cost-effective way to facilitate collaboration between development and project management teams.

In order to integrate a non-functional perspective into DevOps concepts, the report of Brunnert (2015) focused on tools, activities, and processes to ensure one of the most important quality attributes of a software system, namely performance. This report outlined activities assisting the performance-oriented DevOps integration with the help of measurement- and model-based performance evaluation techniques. The report explained performance management activities in the whole lifecycle of a software system and presented corresponding tools and studies. Following a general section about existing measurement- and model-based performance

evaluation techniques, the report focused on specific activities in the development and operation phases. Afterwards, it outlined activities during the evaluation phase when a system is going back-and-forth between development and operation.

Another research study reported by Perez et al. (2015) addressed the problem of the feedback that should be provided to the developer about the performance, reliability and general quality characteristics of the application at runtime. It also referred to what measurement information should be carried back from runtime to design-time. In this paper, the authors described the design of a tool, referred to as the filling-the-gap (FG) tool, to enhance and automate the delivery of application performance information to the developer. The FG tool has two main objectives: The first objective is to provide the data to parameterize application design-time quality of service (QoS) models for improving their accuracy by relying on the monitoring information collected at runtime. The second objective of the tool is to provide developers with a report of the application behavior at runtime. The FG tool presented relies on a monitoring framework capable of providing both application and system-level monitoring metrics.

Wettinger et al. (2014) mention that to improve short release cycles of software, certain technical and non-technical challenges should be tackled. Also, further conditions are required to enable fast and continuous delivery software, such as the split and barrier between developers and operations personnel. This paper focuses on enabling DevOps in combination with cloud computing to reveal the full potential of DevOps and tackle the challenges. A specific form of language is proposed (DevOpSlang) in conjunction with a methodology to implement DevOps as an efficient means for collaboration and automation purposes. Future plans of the authors include implementation of mechanisms to generate DevOps file skeletons based on existing descriptor files, evaluate DevOps files automatically and make suggestions how to improve a given DevOps file.

As previously mentioned, the work by Peuraniemi (2014) provides a clarification to the definition of DevOps and reviews its principles, methodologies and tools, while it summarizes methodologies and tools in IBM.

3.3 DevOps Application

Nybom et al. (2016) deal with the DevOps definition problem. As can be extracted from this work, DevOps has a clear purpose; however, the term is an ambiguous concept and is lacking a standard definition. Organizations adopting DevOps must interpret and define what DevOps means to them. According to the authors, the fact that DevOps is lacking a standard definition implies that there is no simple approach to follow when adopting DevOps in an organization. The paper also presents results from a case study in which a software organization adopts DevOps. The focus of this research was to study the impact of mixing responsibilities between development and operations engineers. To this end the authors interviewed 14 employees in the pilot organization

during the study, and results indicated several benefits of the chosen approach, such as improved collaboration and trust and smoother work flow. As the authors state, this comes at the cost of several complications, such as new sources for friction among the employees and risk for holistically sub-optimal service configurations.

Kerzazi and Adams (2016) described an issue related to practical aspects faced by DevOps engineers. The authors note that despite an increasing number of conferences and publications on continuous delivery, smaller companies and startups still struggle to determine the core tasks in future releases that DevOps engineers should be responsible for, while universities are not sure what essential techniques and skills they should teach to their students. This article performed an empirical study on online job ads to learn from existing companies' job requirements for three specific roles: build, DevOps and release. Through a mixture of approaches, the authors identified 16 core activities manually looked for in the sampled job ads. They found that automation is the most important activity across the three roles and that release engineering job ads seem to combine the top activities of build and DevOps engineers. Stillwell and Coutinho (2015) described how developers on the HARNESS (Hardware- and Network-Enhanced Software Systems for Cloud Computing) project may address this issue through an approach based on version-controlled configuration management, automated software deployment and continuous integration.

The goal of providing continuously updated services should make sure that the overall enterprise performance and security posture are not compromised, while the quick turnaround capability deployment is achieved. Farroha and Farroha (2014) proposed and developed a framework that focuses on ensuring the continuity of strategic posturing while allowing maximum flexibility to tactical enhancements to meet emerging demands.

Schneider and Wolfsmantel (2016) comment that the connected car business has high demands on the exchange of data and files between the connected car on the road, and a variety of services in the backend. They conclude that to solve current and upcoming challenges, a scalable and flexible architecture and team setup is needed. To this end, they proposed the selection of microservices as an architectural paradigm to enable replicating granular services for scalability and to easily replace a depreciated service. In this approach, one team was used for the development and operations of the services, which was responsible for the backend infrastructure in the cloud. Also, a DevOps culture was established allowing quickly deployment of services with increased operational efficiency and code quality.

Borgenholt et al. (2013) proposed Audition, which is a tool to automate complex and work-intensive analysis process. As the authors claim, Audition meets the given performance requirements of the service at the minimum cost. The cloud paradigm already provides cost benefits in terms of the ability to scale resources based on demand. Audition is able to augment the scaling and provide additional cost benefits by minimizing the cost for the current constellation of virtual machines given the current performance requirements.

Wettinger and Andrikopoulos (2015) state that because DevOps automation approaches and cloud services appear, change and disappear so rapidly, it is hard to choose the most appropriate solutions and combinations thereof to implement holistic DevOps automation for a specific application. Informed decision making is also hard because of the huge variety of options. DevOps knowledge is practically available at large scale, but it is not systematically captured and managed. In this work, the authors presented a collaborative and holistic approach to capture DevOps knowledge in a knowledge base. Besides the ability to capture expert knowledge and utilize crowdsourcing approaches, they implemented a crawling framework to automatically discover and capture DevOps knowledge. Moreover, they showed how this knowledge may be utilized to deploy and operate cloud applications.

Many organizations are trying to adapt to the increase of web applications that depend on the cloud. To achieve releasing the product in time, it pressurizes functional teams of the organization to perform as needed. As perfection is rare to achieve, it is natural that more and more defects are produced and consequently this causes annoyance to the team. Mullaguru (2015) discussed the new paradigm of DevOps, which tries to address the identified problem by integrating development and operations teams. In this model, the development team takes care of operational requirements like deploying scripts, debugging and performance testing from the scratch, while the operations team takes care of well-informed support and feedback before, during and after deployment.

Moltchanov and Rocha (2015) report that the classic operations of cloud environments are to add virtual machines when an application is running under resources or releasing these resources when they are not used anymore. Additionally, service provision, where the entire lifecycle of the application or service is under the control of a cloud management subsystem, can be performed with automatic and autonomous start, stop, and restart of a server or an application. However, little effort has been devoted to support a developer during the application development process or during the service execution by changing the application behavior depending on the response and availability of service components. To tackle this problem, the CloudWave project was developed, which was intended to refine modern cloud infrastructures and tools by enabling agile development and delivery of adaptive cloud services; the latter are able to dynamically adjust to changes in their environment so as to optimize service quality and resource utilization. The project emphasized on key concepts such as feedback-driven development (FDD) and the coordinated adaptation (CA) during the development or execution of a service or application.

Bruneo et al. (2014) stated that the engineering methods and tools used to develop cloud services do not cover the needs for faster innovation and improvement of business agility. Thus, the authors note that there is need to increase the speed of innovation by improving agility in designing software and operating cloud based services, as well as for improving cloud infrastructure adaptivity and the interaction between the higher-level software and service goals with the lower-level infrastructure. In this paper, an attempt was described to address this

challenge by delivering novel technologies and methods for improving both the development of Software as a Service (SaaS) solutions and the management of their operation and execution. This new development and management paradigm was called CloudWave and was built upon new technologies and open standards, while it aimed at leveraging assets and outcomes of relevant FP7 EU projects such as RESERVOIR, FI-WARE, Optimis and others. The development of CloudWave was closely accompanied by experimentation and verification of results using real-world industrial use-cases as benchmarks. A reference architecture was created where all functional specification, interface descriptions and portions of the code is released as open source.

The task of managing large installations of computer systems presents many unique challenges related to heterogeneity, consistency, information flow and documentation. A common sight at universities is a highly complex and heterogeneous computer system infrastructure with low turnaround times in terms of both hardware and personnel. Ensuring a high quality of service is mandatory for both teaching and research, but many institutes find themselves hard-pressed for manpower. Schaefer et al. (2013) provide an insight to how automation can to improve scalability and testability, while simultaneously reducing the operators' work. They introduced an approach that makes use of configuration management and systems monitoring, which together enable the operators to test changes in an isolated environment before rolling them out and to track failures in near real time.

Moore et al. (2016) argue that to choose the right technologies to build an urban-scale Internet of Things (IoT) system, there is often a focus on low-level architectural details, such as scalability of message handling. But to build an IoT system requires a high-level holistic approach that mixes traditional data collection from vendor-specific cloud backend, together with data collected directly from embedded hardware and mobile devices. Supporting this heterogeneous environment can prove challenging, leading to complex systems that are difficult to develop and deploy in a timely fashion. This paper described how to address these challenges by proposing a three-tiered DevOps model, which was used to build an information system capable of providing real-time analytics of electric vehicle (EV) mobility usage and management within a smart city project.

Olszewska and Waldén (2015) mention that formal methods are used to develop high-criticality complex systems to guarantee that the system will be correct by construction. However, these methods are heavyweight and also the development requirements for faster delivery and improved communication within the development team and others have changed. So these need to be integrated with other methods to meet new development requirements (namely, agile development). To tackle the issue, a combination of formal and agile software engineering approaches was proposed, which consisted of setting up a Scrum-based process for the development of systems of high criticality with the use of Event-B. The authors also present a merging of formal modelling with Scrum under a DevOps perspective and discuss how formal

modelling can function within DevOps so as to contribute to its quality assurance and continuous delivery.

Cukier (2013) suggests that scaling web applications can be easy for simple CRUD software running when PaaS is used. Scaling of a web application with many components and users requires a mixture of cloud services in PaaS, SaaS and Infrastructure as a Service (IaaS) layers, as well as knowledge in architecture patterns to make all these software components communicate. In this article, the author shared the experience of using cloud services to scale a web application. He showed usage examples of load balancing, session sharing, e-mail delivery, asynchronous processing, logs processing, monitoring, continuous deployment and real-time user monitoring (RUM). These are a mixture of development and system operations that improve application's availability, scalability and performance.

Zheng et al. (2016) describe an attempt of a DevOps team to adopt agile methodologies during a summer internship programme as an initiative to move away from the waterfall lifecycle model. The DevOps team implemented the Scrum software development strategy to create an internal data dictionary web application. This article reported on the transition process and lessons learned from this pilot programme.

Babar et al. (2015) report that although at present applying DevOps to data analytical system (DAS) is increasingly embraced, the characteristics of this system, such as data protection, always leads to a series of constrains. Additionally, the paper claimed that it is hard to conduct DevOps on a data analytical system, and that there are no DevOps solutions for reference. Therefore, exploring DevOps for data analytical system is indeed valuable. This paper, illustrated DevOps demands of data analytical systems from different perspectives, and constantly emphasized the importance of automation toolchain. Based on those, a process model for DAS DevOps (D2Ops) was proposed to clarify participants' activities. Six generic process components were designed to support this model considering also stability, which can be also used as selection criteria for specific automation tools. The paper also presented a reference facility based on these generic process components and illustrated its implementation with a practical case.

Babar et al. (2015) claim that more and more enterprises rely on software for the development and delivery of appropriate products and services; therefore, software processes have become an integral part of enterprise processes. The software development processes can vary significantly from organization to organization due to unique enterprise characteristics. These processes can be reconfigured in multiple ways to take account of enterprise variations and behavioral peculiarities so as to fulfill high-level enterprise requirements. However, current methods of modeling software process reconfigurations are limited in their ability to consider multiple enterprise perspectives and help choose among alternate configurations. In this paper, the authors considered the possible dimensions of software process reconfigurability using the DevOps approach as a motivating example. Limitations of current process modeling languages, such as Business Process Model and Notation, in illustrating multiple aspects of process architecture were discussed, with the Business Process Architecture modeling technique being

used to describe four dimensions of Process Element positioning, namely, temporal, recurrence, plan-execute, and design-use, in a typical DevOps implementation. Goal models were used for evaluating alternate software process reconfigurations by assessing the satisfaction of enterprise non-functional requirements.

Cois et al. (2014) report that a fundamental key to have a successful project is the efficient and effective dissemination of information between the project teams. As information changes or new information becomes available, existing information repositories and artefacts must be updated immediately to avoid injecting defects into the software being developed and wasting effort spent working with incorrect information. Dissemination of information in near real time can take a heavy toll of project resources. This can dramatically reduce productivity of project teams because it takes a large volume of project effort. The authors framed this challenge as a communications problem that can be addressed by the introduction of specifically designed autonomous system actors and processes. Successful implementation of such a methodology may thus enable efficient, effective, and immediate data collection, synthesis, and transfer of information between all requisite entities within the software project. A generalized model of DevOps was presented and analyzed in this paper, which offers a formalization of the communications and actors requisite to any effective software development process. These concepts were further developed to illustrate the information flow between human and system actors, and explore how this model can be used to optimize the processes of a software development team to maximize productivity and quality of work products.

Meirosu et al. (2015) mention that network service providers are facing challenges for deploying new services mainly due to the growing complexity of software architecture and development process. Moreover, the recent architectural innovation of network systems such as Network Function Virtualization (NFV), Software-defined Networking (SDN), and cloud computing increases the development and operation complexity yet again. To tackle this problem, the DevOps concept is used by the companies. The authors argue that although the goals of DevOps in data centers are well suited for the demands of agile service creation, additional requirements specific to the virtualized and software-defined network environment are important to be addressed from the perspective of modern network carriers. This paper also discussed the DevOps requirements specific to the modern network service creations that follow a trend towards employing virtualization and software-defined network technologies. Furthermore, it proposed an extended DevOps concept (SP-DevOps) that addresses the discovered requirements and demonstrated how it can be used in a large-scale service creation by taking the EU FP7 project UNIFY as a reference architecture. In order to further explore the concept of SP-DevOps, the authors described the four main processes, that is, VNF developer support, verification process, observability process, and troubleshooting process, that together form the SP-DevOps lifecycle in a UNIFY service creation. Finally, the paper explained how each of these processes effectively tackles the corresponding DevOps requirement for a large-scale service creation on virtualized and software-defined network environments.

The motivation of Wettinger et al. (2016) was based on the fact that a huge number and variety of reusable DevOps artefacts are currently being shared as open-source software, such as Chef cookbooks and Juju charms. However, these artefacts are usually bound to specific tools such as Chef, Juju or Docker, something that makes it hard to combine and orchestrate artefacts of different kinds in order to automate the deployment of cloud applications consisting of different middleware and application components. In order to tackle this issue, the authors worked on an automated transformation framework to generate TOSCA standard-based modeling artefacts from different kinds of existing DevOps artefacts. First they presented an initial classification of DevOps artefacts, distinguishing between node-centric and environment-centric artefacts. They then presented a generic transformation framework used to transform arbitrary DevOps artefacts into TOSCA node types and relationship types. These can then be used to create topology templates for cloud applications. TOSCA enables the seamless and interoperable orchestration of arbitrary artefacts. Based on their transformation framework, they implemented technical transformation methods for two different kinds of DevOps artefacts, namely Chef cookbooks and Juju charms. Finally, they evaluated their framework, as well as the technical transformation implementations, based on a specific scenario.

Economou et al. (2014) state that DevOps teams typically implement practices summarized by the colloquial directive to “eat your own dogfood" meaning that software tools developed by a team should be used internally rather that thrown over the fence to operations or users. The authors presented a brief overview of how DevOps techniques bring proven software engineering practices to IT operations. They also discussed the application of these practices to astronomical observatories.

Liming et al. (2016) discuss what DevOps is and how its practices can affect developers throughout the software development lifecycle. The article also talks about microservices and gives a brief description of their use. Also, it defines the domains that might benefit from DevOps and the kind of practices that are best to adopt according to the type of systems and organizations.

Ebert et al. (2016) give another form of explanation of what DevOps is and describes some of its tools, while Bang et al. (2013) describe the four perspectives of DevOps: a culture of collaboration between all team members; automation of build, deployment and testing; measurement of process, value, cost and technical metrics; and sharing of knowledge and tools. IT practitioners have used configuration management tools to support DevOps. However, the current DevOps approach is limited to the usage of proprietary configuration tools focused on automation of operations. The authors proposed a new approach to support DevOps; Knowledge, Skills and Abilities (KSA) for DevOps. KSA has been used by the Department of Defense in the U.S. to identify the better candidates from a group of people qualified for a position. For this purpose, the paper first described grounded theory in the context of several KSAs that have been used to develop and deploy modern web applications and apply it to the artefacts of modern web application development projects. Based upon this analysis, the paper

discovered next how the KSAs can support the four perspectives of DevOps and argued that a new theory – modern web application development with a KSAs approach – supported the four perspectives of DevOps.

As previously reported, Wettinger et al. (2014) argue that certain technical and non-technical challenges should be faced to improve short release cycles of software. The target of this paper is to investigate how to enable DevOps in combination with cloud computing to reveal the full potential of DevOps and tackle the challenges. In this attempt the DevOpSlang language is proposed in conjunction with a methodology to implement DevOps as an efficient means for collaboration and automation purposes.

This technical report by Duvall (2012) describes the nascent topics of DevOps and continuous delivery while enumerating the Challenges IT organizations are seeking to remedy by adopting these approaches. Aimed at technology, executives and directors and those responsible for delivering features and stable software systems to their users, the report covers the benefits of a DevOps mindset, which encourages communication and collaboration by obliterating the silos that impede projects. Additionally, it cites the value provided to the business itself, how the approach works, and some of the components and tools that are used on projects to deliver new features to users in a stable environment. Finally, it touches upon the concept that the future will involve less friction and more value from tools and infrastructure that support this approach.

The technical report by Azoff (2011) states that DevOps originated with operations as a means of streamlining and improving the effectiveness of operations in the face of increasing workload. Traditionally operations had the time to deal with application stability, risk, and performance issues separately from infrastructure management and procurement tasks. This changed with the adoption of agile practices in development at one end, resulting in increased deployment frequency, while at the operations end the trend towards cloud and virtualization sped up and lowered costs relating to hardware procurement. Consequently, operations became the bottleneck. The availability of new deployment solutions has significantly helped operations by automating many manual operations. The solutions from a range of leading vendors and innovators in release management and automation are compared side-by-side in the Ovum rainbow map for DevOps. This solution guide and comparison will help IT managers choose a good fit for their operations' needs. Ovum has researched the leading players in release management automation (an area that is receiving renewed focus), and conducted a comparative study of these solutions. A DevOps features matrix was created, and the resulting comparison showed that Electric Cloud, HP, IBM, Serena, and UrbanCode have the most comprehensive coverage of the features desired in advanced DevOps solutions. The areas that were most lacking among solutions were build and performance testing, with vendors typically relying on third-party solutions to offer these features.

Finally, Roche (2013) discusses some DevOps concepts and explains how DevOps affects quality assurance duties and responsibilities.

3.4 Literature Reviews

Erich et al. (2014b) claim that research about DevOps is still in its infancy despite its widespread adoption in industry. Also, they report that the DevOps approach is relatively recent and fragmented, something which makes it hard to see and understand its scope, the topics it covers and the challenges addressed, as well as those remaining unaddressed. The authors performed a systematic literature review on DevOps aiming at helping researchers and practitioners to define it clearly. The review is performed in the context of a larger project, which also includes interviews with industry professionals while the selection of papers may be biased as only English papers were included. The key element was that no universal term for the intersection between development and operations exists, while it also became clear that there was a lot of difference in how the papers studied defined DevOps.

Lwakatare et al. (2015) observe that that currently there is lack of common understanding of what DevOps constitutes in academia and in the practitioners' communities. The authors conclude that there is need for research that investigates the DevOps phenomenon and examines how it impacts software development and operations. This paper identified the basic dimensions of DevOps using relevant academic literature and interviews with practitioners actively involved in the DevOps movement. The main contribution of the study was the definition of the main elements that characterize the DevOps phenomenon and an initial conceptual framework that describes it.

Erich et al. (2014c) provide a discussion about DevOps concepts that are not frequently met in academic literature. The authors selected which papers to consider in their review by using the terms "DevOps", "continuous delivery" AND "software", "development and operations" AND "software". The results provided 14 journal articles, ten conference proceedings and two industry reports out of the 139 articles found. From the journal articles, ten originated from the Cutter IT Journal, which released a special issue about DevOps. The most important finding of this review was that there is no DevOps process or methodology. DevOps is not a one-size-fits-all solution to solve a problem in software engineering, like Scrum for example. DevOps should be considered as an artefact adapted to match the unique environment of an organization under study.

Erich et al. in another paper (2014a) argue that some people believe that the structural division of information system departments into development and operations may have a negative impact to the corresponding supporting processes. The authors attempted to find empirical evidence that DevOps does indeed benefit development and performed a systematic mapping to explore DevOps, resulting in selecting 26 articles out of 139 that were studied and summarized. The main outcome was that DevOps has not been adequately studied in scientific literature. However, DevOps benefits development and operations performance, while it also has positive effects on web service development and quality assurance performance.

3.5 Research Challenges

This section outlines the most significant research challenges reported in the papers studied.

Shahin (015) claims that DevOps/CD (continuous deployment) brings new challenges for software architects, which have considerable impact both on their (architectural) design decisions and on their organizational responsibilities. However, the author states that there has been a little research devoted on what implications DevOps/CD can have on architecting and that there is an important and urgent need to gain a deep understanding of how DevOps/CD adoption can influence the architecting processes and their outcomes in an organization. This is conducted to seek for architectural elements, practices and patterns that support DevOps and continuous deployment. The research methods used were systematic literature reviews, exploratory case studies, data collection methods and data analyses. This is an open research challenge which should be pursued further.

Meirosu et al. (2015) open a discussion in the Network Function Virtualization Research Group (NFVRG) about challenges related to transforming the telecom network infrastructure into an agile, model-driven production environment for communication services. Their inspiration was taken from a DevOps data center aiming to simplify and automate management processes for a telecom service provider software-defined infrastructure (SDI).

Hamunen in his thesis (2016) studies the challenges of DevOps by interviewing nine experts who were involved with DevOps initiatives in their companies. The findings were divided into four main challenge categories based on their topic: lack of awareness in DevOps, lack of support for DevOps, implementing DevOps technology and adapting organizational processes to DevOps. The thesis suggests that clearing misconceptions and spreading the knowledge of DevOps helps overcome the lack of awareness challenge. Additionally, building commitment and trust in both management and team-levels makes DevOps more appealing for an organization. Establishing common ways of working and leading by example help also to overcome the challenge of fragmented technologies and reluctant attitudes towards it. Finally, ensuring the flexibility of the organization is the key to prevent bottlenecks in the delivery process.

4. Identification of gaps and unexplored areas

Based on the findings presented in the previous sections, the first identified gap that immediately arises is the lack of significant academic research considering the wide acceptance of the DevOps phenomenon in the industry. This gap has already been identified by other review studies regardless of their scope (Erich et al., 2014a; Hamunen, 2016; Lwakatare et al., 2015). In this

section we cite weaknesses, limitations and future research steps that may be conducted based on the findings of the collected research works and following the same categorization rationale.

DevOps Application

Bang et al. (2013) focus on further research work that is needed to discover how we can educate and train software developers and operators to be equipped with knowledge, skills and abilities to support DevOps. This is quite critical as education in DevOps will certainly open the doors for a wider adoption by both the research community and practitioners.

The findings by Lwakatare et al. (2016) show that the DevOps phenomenon is more prominent in organizations providing services over the Internet. It would be beneficial for future research to focus on further empirical evidence of DevOps practices and patterns in companies that claim to have implemented it. More important will be research that not only identifies the practices but also considers the kinds of system, organizations and domains in which DevOps practices are applicable.

Wettinger and Andrikopoulos (2015) suggested that the management of the huge available DevOps knowledge can be improved by developing and populating a DevOps knowledge base with more offerings from cloud providers, the evaluation of crawling techniques for the automatic capturing of such offerings (e.g., based on the providers' API documentations) and, finally, by enabling the automated generation of alternative application topologies based on previous established research.

Challenges

When it comes to research challenges and open problems, it becomes obvious that the lack of academic research in the field of DevOps has a negative effect on describing and pursuing such endeavors. Literature reviews and related studies reveal that the majority of the sources used are non-academic (such as those used by Hamunen (2016)). The means that any attempt to record, as well as to overcome DevOps-oriented challenges and especially in the area of Software Engineering are based on the experiences of practitioner. Further research can be made by expanding the amount of respondents and carrying out scientific research such as a quantitative analysis. A larger pool with research material could give a better overall picture of the challenges and contribute to defining which of them the most common ones are.

Literature Reviews

Erich Floris and his group of collaborators lead the way in conducting literature reviews and identify weaknesses and gaps. Some of these are outlined below.

In their work they state that further research could benefit from a clarification of DevOps by creating a taxonomy using the culture, automation, measurement, sharing (CAMS) framework (Erich et al., 2014b). Also, they suggest that an extended version of the CAMS framework,

including the concepts of services, quality assurance and structures and standards, can be used as a framework for classifying DevOps research.

In another study of the same group, the authors observed weaknesses and bias in the paper selection process and the publication of only positive results (Erich et al., 2014a). More specifically, the inclusion of English-only papers may possibly exclude relevant studies in other languages actively exploring DevOps. Publishing only positive results might exclude reports by organizations that struggle with DevOps and might have abandoned it. Further research is needed to discover whether DevOps actually increases is development and operations.

Finally, in another study the group conclude that relatively little research is available on DevOps and the low quality of some of the corresponding papers prevents literature studies to come up with specific and concrete results (Erich et al., 2014c).

Tools & Methodologies

This subarea is heavily dependent on practitioners and so empirical investigation could significantly benefit the relevant findings. Nevertheless, it is observed that either the studies are limited in this respect or involve a limited number of stakeholders, like in the study of Liu et al. (2014), which is confined only to one firm from a particular industry sector. Therefore, more research is required in this respect and future efforts should concentrate on these aspects.

As Brunnert et al. (2015) indicate, most of the model-based approaches are developed in the academic environment and not in the industry domain. Further validation on large industry projects would increase the level of trust and readiness to assume the costs and risks of applying performance models. Both industry and academia have to address these challenges to enable a fully performance-oriented DevOps integration. Therefore, it is once again revealed that a close collaboration between researchers and practitioners is of paramount importance to delivering useful tools and methodologies from the academia to the industry.

Another DevOps related gap is identified by Peuraniemi (2014). Peuraniemi takes into consideration only one aspect for improving delivery and feature flow, while monitoring and metrics were not taken into consideration. This partial implementation showed that large companies can benefit from the DevOps approach; however it is hard to determine whether results would be as good when all DevOps principles are to be implemented. Also, when viewed from a large organization perspective this case study may be classified as small, not giving a clear picture on how DevOps principles would affect the organization. Thus, in alignment with what has previously been stated, more studies in collaboration with the industry are needed, which will also address scalability issues and differences in the working environment so that the outcomes and conclusions of such studies can be generalized.

5. Conclusions

The DevOps approach appears to be one of the major trends in the software industry which brings a new culture between development and operations teams aiming to establish a closer collaboration and communication without silos and barriers between them. This report aimed to present a literature overview covering the most significant issues on DevOps approaches related to software engineering methodologies and practices by considering strictly academic research. Following a specific methodology, we identified a set of relevant articles and tried to address two specific research questions: RQ1 - What is the scientific research devoted to the DevOps approach? RQ2 - Which are the main aspects of recent studies on DevOps approach?

It became clear that there is a huge diversity in the approaches used to address fundamental issues or specific DevOps problems. Literature is not clear or consistent as regards the definition of the DevOps phenomenon, while the scientific or research studies are scattered and limited. Most of the material that deals with the DevOps approach comes mostly from practitioners and reflect personal experiences and views that apply to the company or organization where the study was conducted. In addition, it becomes clear that there is great need for tools, methodologies and, in general, applied research in the DevOps area, starting from defining with a clear and unambiguous way its key terms and concepts, and reaching to introducing methods and techniques, and/or implementing tools and processes for supporting it.

The two most important findings of this report are the lack of academic research on DevOps despite the great interest from the industry and the fact that there is no specific common framework or concrete definition for the DevOps approach.

The increasingly growing acceptance and adoption of DevOps approach by the industry, in conjunction with the lack of concrete definitions and limited scientific research, are expected to eventually attract academia's research interest and turn DevOps into a major subject of study. This implies that a continuous research attempt is about to be initiated in the near future that will cover gaps and address weaknesses in the field, offering DevOps oriented software engineering those solutions that will address all relevant challenges.

References

Ahmadighohandizi, F., & Systä, K. (2015). ICDO: Integrated cloud-based development tool for DevOps. *SPLST*.

Akshaya, H., Vidya, J., & Veena, K. (2015). A Basic Introduction to DevOps Tools. *Citeseer*.

Azoff, M. (2011). DevOps: Advances in release management and automation.

Babar, Z., Lapouchnian, A., & Yu, E. (2015). Modeling DevOps deployment choices using process architecture design dimensions. *IFIP Working Conference on The Practice*.

Bang, S., Chung, S., Choh, Y., & Dupuis, M. (2013). A grounded theory analysis of modern web applications: knowledge, skills, and abilities for DevOps. *Proceedings of the 2nd Annual*.

Borgenholt, G., Begnum, K., & Engelstad, P. (2013). Audition: a DevOps-oriented service optimization and testing framework for cloud environments. *Norsk Informatikkonferanse (NIK)*.

Bruneo, D., Fritz, T., & Keidar-Barner, S. (2014). CloudWave: Where adaptive cloud management meets DevOps. *(ISCC), 2014 IEEE …*.

Brunnert, A., Hoorn, A. van, Willnecker, F., & Danciu, A. (2015). Performance-oriented DevOps: A research agenda. *arXiv Preprint arXiv:*

Cois, C., Yankel, J., & Connell, A. (2014). Modern DevOps: Optimizing software development through effective system interactions. *Conference (IPCC), 2014 IEEE …*.

Cukier, D. (2013). DevOps patterns to scale web applications using cloud services. *Proceedings of the 2013 Companion Publication for*.

Duvall, P. (2012). Breaking down barriers and reducing cycle times with DevOps and continuous delivery. *Retrieved from GigaOM Pro Website: Http://try. Newrelic.*

Dyck, A., Penners, R., & Lichter, H. (2015). Towards definitions for release engineering and devops. *Of the Third International Workshop on …*.

Ebert, C., Gallardo, G., Hernantes, J., & Serrano, N. (2016). DevOps. *IEEE Software*.

Economou, F., Hoblitt, J. C., & Norris, P. (2014). Your data is your dogfood: DevOps in the astronomical observatory.

Edwards, D. (2010). What is DevOps. *Viitattu*.

Erich, F., Amrit, C., & Daneva, M. (2014a). A mapping study on cooperation between information system development and operations. *International Conference on Product-Focused*.

Erich, F., Amrit, C., & Daneva, M. (2014b). Cooperation between software development and operations: a literature review.

Erich, F., Amrit, C., & Daneva, M. (2014c). Report: Devops literature review. *University of Twente, Tech. Rep*.

Farroha, B., & Farroha, D. (2014). A framework for managing mission needs, compliance, and trust in the devops environment. *Military Communications Conference*.

Forsgren Velasquez, N., Kim, G., Kersten, N., & Humble, J. (2014). State of DevOps report. *Puppet Labs and IT Revolution*. article.

Hamunen, J. (2016). Challenges in adopting a Devops approach to software development and operations.

Humble, J., & Molesky, J. (2011). Devops: A software revolution in the making. *Cutter IT Journal*, *24*(8), 6–12. article.

Kerzazi, N., & Adams, B. (2016). Who needs release and devops engineers, and why? *Continuous Software Evolution and*.

Kim, J., Meirosu, C., Papafili, I., & Steinert, R. (2015). Service provider DevOps for large scale modern network services. *(IM), 2015 IFIP/IEEE …*.

Liu, Y., Li, C., & Liu, W. (2014). Integrated solution for timely delivery of customer change requests: A case study of using devops approach. *International Journal of U-and E-Service, Science and*.

Lwakatare, L. E., Kuvaja, P., & Oivo, M. (2015). Dimensions of DevOps. In *International Conference on Agile Software Development* (pp. 212–217). inproceedings.

Lwakatare, L., Kuvaja, P., & Oivo, M. (2016). An Exploratory Study of DevOps Extending the Dimensions of DevOps with Practices. *ICSEA 2016*.

Meirosu, C., Manzalini, A., Kim, J., & Steinert, R. (2015). DevOps for software-defined telecom infrastructures. *TASK FORCE (IETF)*.

Moltchanov, B., & Rocha, O. (2015). CloudWave smart middleware for DevOps in Clouds. *Conference on Smart Spaces*.

Moore, J., Kortuem, G., Smith, A., & Chowdhury, N. (2016). DevOps for the Urban IoT. *Proceedings of the*.

Mullaguru, S. (2015). Changing Scenario of Testing Paradigms using DevOps–A Comparative Study with Classical Models. *Global Journal of Computer Science and*.

Nybom, K., Smeds, J., & Porres, I. (2016). On the Impact of Mixing Responsibilities Between Devs and Ops. *International Conference on Agile Software*.

Olszewska, M., & Waldén, M. (2015). DevOps meets formal modelling in high-criticality complex systems. *Proceedings of the 1st International Workshop*.

Perez, J., Wang, W., & Casale, G. (2015). Towards a devops approach for software quality engineering. *Proceedings of the 2015 Workshop on*.

Peuraniemi, T. (2014). Review: DevOps, value-driven principles, methodologies and tools. *Data-and Value-Driven Software Engineering with …*.

Roche, J. (2013). Adopting DevOps practices in quality assurance. *Communications of the ACM*.

Schaefer, A., Reichenbach, M., & Fey, D. (2013). Continuous integration and automation for DevOps. *IAENG Transactions on Engineering*.

Schneider, T., & Wolfsmantel, A. (2016). Achieving cloud Scalability with Microservices and DevOps in the Connected Car Domain. *Software Engineering (Workshops)*.

Shahin, M. (2015). Architecting for devops and continuous deployment. *Proceedings of the ASWEC 2015 24th Australasian*.

Stillwell, M., & Coutinho, J. (2015). A DevOps approach to integration of software components in an EU research project. *Proceedings of the 1st International Workshop*.

Waller, J., Ehmke, N., & Hasselbring, W. (2015). Including performance benchmarks into continuous integration to enable DevOps. *ACM SIGSOFT Software*.

Walls, M. (2013). *Building a DevOps culture*.

Wettinger, J., & Andrikopoulos, V. (2015). Automated capturing and systematic usage of devops knowledge for cloud applications. *(IC2E), 2015 IEEE …*.

Wettinger, J., Breitenbücher, U., & Kopp, O. (2016). Streamlining DevOps automation for cloud applications using TOSCA as standardized metamodel. *Future Generation*.

Wettinger, J., Breitenbücher, U., & Leymann, F. (2014). DevOpSlang–bridging the gap between development and operations. *European Conference on*.

Zheng, J., Liu, Y., & Lin, J. (2016). Exploring DevOps for Data Analytical System with Essential Demands Elicitation. *SEKE*.

Liming, Z., Bass, L., & Champlin-Scharff, G. (2016). DevOps and its practices. *IEEE Software*.