An Exploratory Study of DevOps
Extending the Dimensions of DevOps with Practices

Lucy Ellen Lwakatare, Pasi Kuvaja, Markku Oivo,

M3S, Faculty of Information and Electrical Engineering
University of Oulu,
P.O. Box 3000, 90014 Oulu, Finland
Email: firstname.lastname@oulu.fi

***Abstract*—Software-intensive companies constantly try to improve their software development process for better software quality and a faster time to market. The DevOps phenomenon emerged with the promise of easing the process of putting new software changes to production at a fast rate whilst also increasing the learning and innovation cycles of their products. However, the DevOps phenomenon lacks clear definition and practices, and this makes it difficult for both researchers and practitioners to understand the phenomenon. In this paper, we focus on consolidating the understanding of DevOps and its practices as described by practitioners using multivocal literature and interviews. The study contributes to a scientific definition of DevOps and patterns of DevOps practices to help identify and adopt the phenomenon.**

Keywords–DevOps; Continuous Deployment; Agile.

I. INTRODUCTION

Innovative online companies, such as Amazon, Google and Facebook, have fuelled customers expectations for great services at fast speed due to their quick response times to customer demands. Consequently, more companies from most fields are learning and emulating their capabilities in order to cope with competition and technological changes in the field of IT [1]. Today's technology landscape and advances, such as cloud computing, have changed the ways in which software products are developed and delivered to customers. For instance, in the cloud environment, providers of Software-as-a Service (SaaS) applications are expected to update software frequently and in much faster release cycles to customers.

The recent paradigm shift towards fast and frequent delivery of software updates to customers is referred to as continuous deployment (CD) [2]. CD has been described as an evolutionary step after Agile and continuous integration (CI) practices [2] [3]. CD is a practice whereby software features and updates are rapidly rolled out to production as soon as they are developed, whilst also rapidly learn from real-time customer usage of software [2] [3]. The advantage is that companies can proactively identify and validate assumptions of customer needs by applying practices, such as feature experimentation, that tightly integrate runtime-monitored data fromEmphsi production into the software development activities [4].

Responsiveness to customer needs achieved through CD can put a strain on functional teams within an organisation [2]. Consequently, the DevOps phenomenon emerged with the aim of breaking down organisational silos and encouraging cross-functional collaboration among stakeholders involved in software development and delivery processes–especially development and IT operations. The DevOps phenomenon, despite its growing interest in software industry, faces several challenges such as the lack of a clear definition [5]. This lack of clear a definition has resulted to a number of problems and criticisms, including tensions as to whether DevOps is about culture, technical solution or, alternatively, an entirely new role within a software development organisation [6].

The goal of this research is to consolidate the understanding of DevOps phenomenon as described by practitioners. We use an exploratory case study technique that involves a review of multivocal 'grey' literature and interviews. Our work extends other previous studies that have tried to characterise the DevOps phenomenon. Multivocal literature review and interviews were selected as appropriate approaches for this study because DevOps is very much driven by practitioners, and as such, contribution from non-scientific community are worthwhile. The contribution of this paper is twofolds. First, to validate and improve the scientific definition of DevOps proposed by Penners and Dyck [7]. Second, to extend our work on the dimensions of DevOps [8] with a set of examplary practices and patterns of DevOps. The following research questions are addressed in this study:

- RQ1: How do practitioners describe DevOps as a phenomenon?
- RQ2: What are the DevOps practices according to software practitioners?

This paper is organized as follows: Background and related work, including a scientific definition of DevOps, are presented in the next section. Section 3 presents our research methodology. The results of the study are presented in Section 4, which is followed by a discussion and conclusions in Section 5 and 7, respectively. Section 6 presents validity threats including limitations of the study.

II. BACKGROUND AND RELATED WORK

According to Humble and Molesky [9], DevOps– a blend of two words Development and Operations– is about aligning incentives of everybody involved in delivering software, with particular emphasis on developers, testers and operations personnel. The problems resulting from misalignment between development and operations are not new, though their appearance in the literature is scarce [10]. Prior studies investigating cooperation between developers and operations personnel in real contexts have revealed that very often development and operational activities are not tightly integrated [10] [11]. The latter, according to Iden, Tessem and Päivärinta [11], results

 ISBN: 978-1-61208-498-5

to a number of problems, including IT operations not being involved in requirements specification, poor communication and information flow between the two groups, unsatisfactory test environments, lack of knowledge transfer between the two groups, systems put into production before they are complete and operational routines not established prior to deployment. These problems were identified from a delphi study consisting of 42 experts grouped in three panels representing the roles of developers, operations personnel and systems owners. In the latter study, the authors [11] concluded that operations personnel are to be regarded as important stakeholders throughout system development activities, especially in systems requirement, testing and deployment.

The closest similar work, though different in terms of the studied phenomenon, is by Tom, Aurum and Vidgen [12]. Using a multivocal literature review (MLR) approach supplemented by interviews, the authors of the latter study [12] examined and consolidated the understanding on an unclear phenomenon i.e., technical debt. The approach was used by the authors to develop a theoretical framework that gives a holistic view of the phenomenon comprising of dimensions, attributes, precedents and outcomes. In this study we apply similar approach used by Tom, Aurum and Vidgen [12] to consolidate the understanding of DevOps phenomenon whilst also compare and complement other related works of DevOps particularly those done by Kerzazi and Adams [6] and Penners and Dyck [7].

A. DevOps Definition and Practices

DevOps is a phenomenon that has often been said to a lack clear and precise definition [5] [7] [8] [13]. Its ambiguity and lack of clarity often hinders its adoption [5]. To address this problem, a scientific definition of DevOps is proposed by Penners and Dyck [7]. The definition was derived from comparing and contrasting various descriptions of DevOps and release engineering as the two terms seemed to have a big overlap [7]. According to the authors, DevOps and release engineering share the same goal of providing high-quality software as fast as possible. However, DevOps tries to achieve the goal by improving the collaboration aspect, whereas release engineering addresses the goal in a holistic way i.e., covers all aspects that impact or influence the delivery process of software product to customer [7]. The authors [7] define DevOps as:

> "*a mindset, encouraging cross-functional collaboration between teams - especially development and IT operations - within a software development organization, in order to operate resilient systems and accelerate delivery of changes* ".

This definition of DevOps was developed through discussion with some experts, but the feedback of its description from practitioners was not as consistent as that of release engineering. Based on this, the authors recommended additional inquiry from more practitioners for a more precise definition of DevOps. This study explores how different practitioners have described DevOps and in addition compare our findings of such descriptions with the definition provided by Penners and Dyck [7] [13].

In addition to the definition of DevOps, different aspects and dimensions characterising the DevOps phenomenon such as culture, sharing, automation, collaboration and measurement have been presented in other works [8] [9] [14]. However, the various aspects or dimensions of DevOps seem to lack a consolidated overview to a set of practices and patterns attributed to DevOps [15] [16] [17]. The latter is largely due to the limited number of scientific studies reporting the DevOps phenomenon in practice, although this is changing as there is presently an increasing number of studies of DevOps e.g., studies from IEEE Software special issue on DevOps [18] [19] [20].

III. METHODOLOGY

We report an exploratory case study [21] on the DevOps phenomenon conducted between September 2015 and April 2016. Exploratory case studies are useful in finding out what is happening on a phenomenon whilst also seek new insights and generate ideas for new research.

A. Data Collection

This study uses qualitative data–both primary and secondary data– collected from practitioners in two phases. Phase 1 involved collecting primary data using semi-structured interviews with software practitioners from one case company. Secondary data, consisting of readily accessible writings from the Internet, such as blogs, white papers, trade journals and articles, were collected in phase 2 of the study using Google search. The documents gathered during Phase 2 are collectively referred to as multivocal literature (ML) [22]. The multivocal literature review provides a method for exploring a common, often a contemporary topic of interest [22]. ML embodies views and voices of diverse set of authors expressed in variety of forms [12] [22].

1) Phase-1 Interviews: Three interviews with software practitioners were conducted in one company (Company A), that provides consultancy and product engineering services to its customers. In the latter company, we focused our study on one project, that involved the development of cloud-based road maintenance reporting tool. The company was developing the tool for a customer in the public sector. Convenience sampling was used to select suitable companies based on their participation in the Need for Speed (N4S) project (http://www.n4s.fi/en/), which the present study is part of. A contact person from company A was asked to select a project team that was applying DevOps practices. All interviews were recorded and later transcribed for analysis. The roles of the interviewed practitioners are summarized in Table I .

TABLE I. SUMMARY OF INTERVIEWEES.

Company	***A***
Number of Employees	< 350
Studied Project	Road maintenance tool.
Team Size	7 (co-located).
Number of Interviews (role)	2 (senior developers) 1 (project manager)

A semi-structured interview guide was used during the interviews and had questions that inquired about the (1) current way of working in software development and deployment, (2) strengths and weaknesses in ways of working and (3)

definitions, practices, benefits and challenges of DevOps and CD. DevOps questions were guided by the results of our previous work that analysed and synthesized the DevOps phenomenon as described in academic literature [7].

2) Phase-2 review of multivocal literature: In phase 2, we conducted MLR in the beginning of March 2016. In this study, we selected the MLR as an appropriate method because DevOps is a phenomenon that is largely driven and discussed by practitioners in non-academic circles, and as such it requests the review of the largely available information in non-scietific forums, as seen also in a study done by Kerzazi and Adams [5]. Despite the apparent issues of rigor associated with MLR approach [22], the contemporary nature of DevOps phenomenon suggests some value in reviewing ML that cannot be overlooked. As noted by Ogawa [22], the information presented in ML cannot be assessed in the same way as that of academic literature. We therefore considered various recommended procedures by Ogawa [22] in order to minimize bias and error that can be transferred to, and incorporated in the review of ML. The latter included focusing our study on presenting a thicker picture of the DevOps phenomenon that would serve as input for more sophisticated examination in future research. We carried out the MLR in the following three stages:

a) Data sources and search strategy: Google web search engine (http://www.google.com) was used to source ML from the World Wide Web. The query used to retrieve results from the search engine was what is DevOps. From the retrieved results, we went through the links provided, page by page, saving the outputs of each link as a PDF file until the page where job adverts started and at this point, the review was stopped. A total of 230 records were collected as data sources and included in subsequent steps.

b) Inclusion and exclusion: A total of 230 documents were imported to NVivo (www.qsrinternational.com/product) for analysis. The inclusion and exclusion of the records were done simultaneously with the initial coding of these records. The process also involved classifying the sources with different attributes, such as author information, e.g. name, role and place of work; and source information, e.g. publication year, forum and link. After reviewing all 230 documents, 201 sources were included as relevant documents and excluded 29 documents as they were either duplicates, video links, pointers to catalogues, course adverts, certification adverts or presentation slides.

c) Quality assessment: Quality assessment was mostly done when classifying the sources with metadata, e.g. author name, role, place of work/affiliation, year of publication, view point. This process offered minimal assessment of position, certainty and clarity of source or alignment with research goal i.e., to consolidate the understanding of DevOps phenomenon as determined by the research questions.

B. Data Analysis

Interview transcripts and results from the MLR (list of ML at http://tinyurl.com/z3jpu5v) were coded in NVivo. At first, codes were assigned inductively to the following categories: (1) Definition (with referred as and description as subcategories) and (2) Practices. Other emerging themes found within and across the sources were also coded (e.g., Motivations for DevOps, DevOps in relation to Agile and CD, Problems addressed by DevOps). Following the inductive coding of raw data into the main categories, i.e., Definition and Practices, a second iteration of inductive coding was performed to form other subcategories. The latter proceeded in multiple iterations until similar patterns in the practices and definition were identified and saturation reached. Interestingly, the practices could be grouped into the dimensions of DevOps identified in previous work [8]; however, a new dimension was also added.

IV. FINDINGS

In this section, we present the findings with respect to the two research questions. Fig. 1 gives a summary of the findings from MLR. For MLR, Internet blogs made up the largest source with 53.2% (107) of all sources. Compared to personal blogs, most of the blogs were affiliated with companies. The MLR showed a growing number of sources writing about DevOps over the past 5 years since its conception in 2009.

From the case company, the project team in company A uses Scrum– agile software development method, with a 3-week sprint cycle. The development team has three environments to which software changes are deployed, i.e., development, test and staging. The production environment was not in use at the time of the interviews because the project was phased and the deliverables as well as the schedule of each phase was determined by the customer. The development team automatically deployed software changes from the test to staging environment using Ansible playbooks. The infrastructure team located in another city supported the development team by provisioning the virtual servers used to create the environments.

A. How do practitioners describe DevOps as a phenomenon? (RQ1)

Most practitioners acknowledge that DevOps is a concept that is difficult to define with accuracy. However, many of them still used different terms and descriptions to describe the DevOps phenomenon as elaborated below:

1) DevOps referred to as: A large number of MLR sources have referred to DevOps as a cultural and professional movement (52). Other terms used to describe the DevOps phenomenon in MLR sources include practices (41), culture (38), approach (38), philosophy, mindset, ideology (37), tool (36), a set of values and principles (30), software methodology (24), role, team or engineer (19) and strategy (8). Fig. 2 summarizes the prevalence of the terms in MLR sources. It should be noted that a single source of MLR can use more than one term to describe the DevOps phenomenon.

2) DevOps descriptions: There is an agreement that the term is a combination of development and operations, that encourages collaboration between software development and operations activities. However, when describing the concept further, practitioners often diverge their descriptions depending on whether they place the emphasis more on the goal or more on the means of achieving collaboration. The most common goals that were described include the following: to reduce response time and fast deployment of high quality and reliable software products and services, to allow instant interactions, to unify workflows for transparency and collaboration. On the other hand, those that emphasise means include: through advanced automation and, by evolving traditional roles of

 ISBN: 978-1-61208-498-5

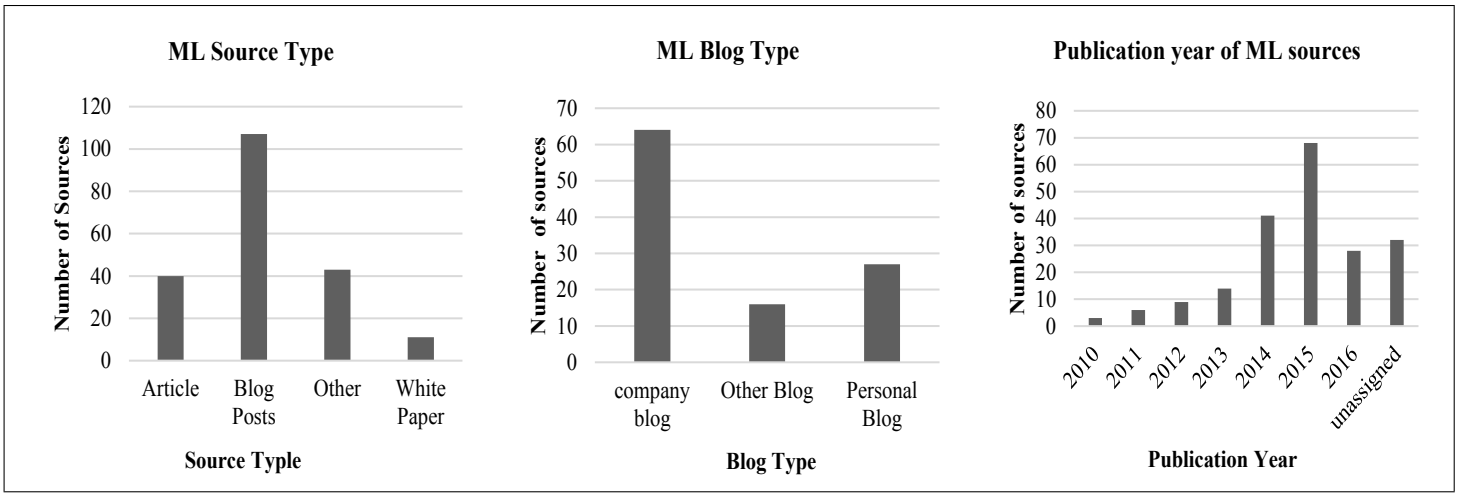

Figure 1. ML Sources.

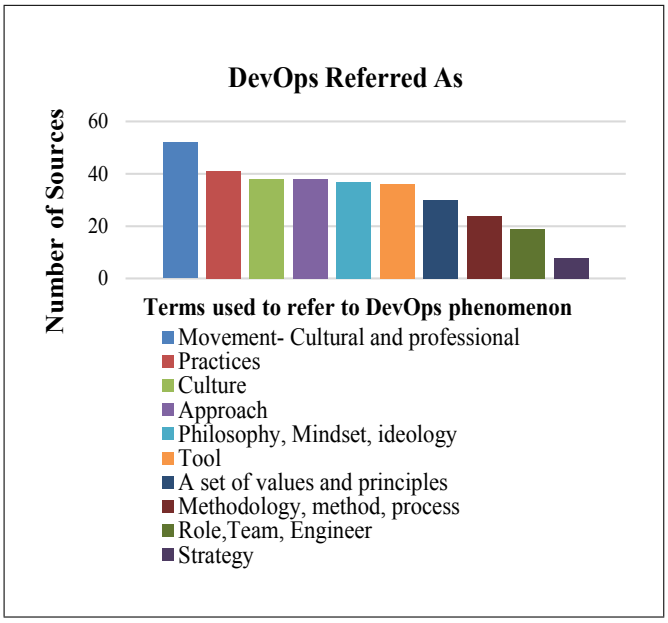

Figure 2. Terms referring to DevOps and their prevalance in ML.

development and operations. In company A, two out of three interviewed practitioners had an understanding of DevOps. One person, the project manager, had no prior understanding of the concept and had learned about it through our study. The following descriptions are extracts from interviewed practitioners in company A when asked how they understood DevOps. The first developer described DevOps as

> *a set of practices to govern everything that is related to installing the software, maintaining it, making sure that all these connections work, i.e. firewalls, version management, etc. It is kind of a little bit of what happens, right after the actual development.*

Another description given by the second developer was:

> *DevOps is mostly about the organisation of work, tearing down the walls that separate typical development organisation from the operational organisation...like instead of provisioning new systems with a ticket, automation projects are modifying the servers and developer can make or commit changes to them by themselves when needed. If you have a separate ops team, then you need to have a very good practice in place for documenting changes and transferring knowledge about the system. A DevOps person will know how to run their own system .*

B. What are the DevOps practices according to software practitioners? (RQ2)

DevOps practices can be crystalized into five dimensions that characterize DevOps. The five dimensions are further elaborated with examplary DevOps practices and patterns to the practices below. Table 2 summarizes some of the DevOps practices in each of the dimensions.

1) Collaboration: rethinking and reorientation of roles and teams in development and operations activities: The issue of role is one of the most widely discussed topics by practitioners in ML. On the otherhand, DevOps phenomenon offers no blue print of how companies can reorganize those roles. A common pattern that was observed is that, companies are forced to rethink and reorient roles, whether new or existing, around the performance of the entire system or service, as opposed to the performance of a specific silo of department or an individual module. Practices in collaboration are often seen as empowerment to team members, especially for the developer since in some cases they gain more control over system operability. This in turn helps to broaden their skillset and knowledge. According to some practitioners, such control allows a single team to be responsible for all aspects, i.e. development and operations of the entire software product or service. Some criticism to this was also observed, including coding time for developers is reduced and the fact that developers need to find ways to effectively balance the support and operations tasks alongside development tasks. Others agree that, companies will need to pick right technology and methodologies to be able to empower developers, as advocated by DevOps. We identified the following two common practices in the collaborative aspect of DevOps.

 ISBN: 978-1-61208-498-5

TABLE II. DEVOPS DIMENSIONS, PATTERNS AND PRACTICES

DevOps Dimension	Patterns of DevOps practices	Examples of practices found in ML
Collaboration	Rethinking and reorientation of roles and teams in development and operations activities	Increasing the scope of responsibilities • Developers paying closer attention to deployment scripts, configuration files, load and performance testing and other activities usually associated with operations groups • Developers learning from operations about resilience, monitoring and diagnosing distributed systems • Developers leverage the support enabled through virtualization, microservice architecture and automation in deployment pipeline to do less operations work, e.g., Docker to remove the need for specifying environments specifications • Development has access and can make changes to critical environments, e.g. production Intensifying cooperation and involvement in each others daily work • Development rotates roles with operations teams, operations attend developer stand-ups and showcases • Operations involved earlier in development to understand what project environments are required to support the application. Also, regular meetings, e.g. weekly to discuss cross-team priorities • In circumstances in which production incidents occur, development and operations come together to troubleshoot and resolve problems as one team • Setting-up shared (virtual and physical) workspace
Automation	Infrastructure and deployment process automation	Infrastructure-as-code • Automate and maintain infrastructure configurations and files using tools such as Chef, Puppet • Developers are shielded from infrastructure issues and are able to create virtual development, test and production environments as well as deploy application using tools like Vagrant and Docker • Scripts used to handle infrastructure are versioned, testable and repeatable • Immutable infrastructure, i.e. artifacts, in their production environments are not updated, rather the infrastructure is always replaced Deployment process automation • Production-like environments are used by development teams for development and testing • Developers self-service environments and deployments • Consistent, reliable and repeatable deployment mechanism across different environments • Configuration changes across environments are automated
Culture	Empathy, support and good working environment between development and operations	• Both developers and operations wear pagers as responsible persons to handle incidents • Integrating development into blameless production post-mortems • Making communication between development and operations non-adversarial and less formal • Mutual respect, support and willingness to work together and share responsibilities
Monitoring	Instrumenting application and aggregating monitored data into insights	• Developers and operations are both involved in determining and implementing monitoring parameters of a system • Set-up monitoring on the production environment for development teams visible through radiators • Development team, including QA, use small subset of high-priority test cases to be executed in production to actively monitor the environment • Effective instrumentation of software by development in collaboration with operations to give information about its health and performance. Also, developers are able to quickly recover code failures in production using aids, such as feature flags
Measurement	Useful Metrics	• Both operations team and development team are incentivized and rewarded by the same metrics • Both development and operations focus on business value as the essential unit of measurement • Progress in development is measured in terms of a working system in production environment • Developers use production feedback to drive decisions, improvements, and changes to the system

a) Increasing the scope of responsibilities: This practice is particularly prominent in the developer role and is the most common implementation of DevOps phenomenon. According to practitioners, developers are responsible for other tasks in addition to designing, coding and testing. Increasing the scope of responsibilities involves pro-activeness of team members in learning the new tasks or alternatively leverage on the automation done to the technical infrastructure of the project. This was also evident from the interviewed practitioners as expressed by the developer from company A: *Shared access to change things, so not just that there's collaboration within the ops teamand ask should you even have an ops team and a development team. If something is broken on the production machine, why do you need an ops team to fix it? It should be possible for anyone in the dev team or any other team to just fix things that are broken.*

b) Intensifying cooperation and involvement in each others daily work: DevOps requires both groups to recognise their key skills in order to share and learn from each other. This aspect may not necessarily mean retraining or crossskilling but encourages providing feedback and visibility across the teams in order to improve. Involving members from each others

teams, development and operations teams can provide valuable information outside individuals areas of expertise. As such, practices such as early involvement of operations in development are emphasized. The following interview extract is a response given by an interviewed practitioner from company A when asked to elaborate how the development team interacts with the infrastructure team:

> *We interact with them through HipChat, which is like an IRC channel. We have a project channel and a member of infrastructure team; [name] has been allocated to support our project. We are basically just sending a message to [name]if we need some help, e.g., we want two more servers to run Java virtual machine and Jenkins, and then he waves some magic and we get what we want. It's pretty instant even if we are technically in a different team i.e., virtual team element, in it.*

2) Automation: infrastructure and deployment process automation: Automation underlies most of the practices that constitute DevOps. Two common practices include automation of deployment process and infrastructure-as-code (IaC). The two practices help to eliminate error-prone manual processes associated with deployments and configuration settings for improved traceability and repeatable workflows. For instance, the use of IaC practice to standardize development, test and production environments.

a) Deployment process automation: as evident in deployment pipeline that views the entire development to operations lifecycle as one coherent end-to-end process rather than numerous multiple and diverse steps. The deployment pipeline is an integrated flow that puts great emphasis on the automation of build, test and deployment processes. It involves continuous development whereby code written and committed to version control is built, tested and installed (deployed) in the production environment. An automated deployment process takes into account and ensures the management of dependencies, versions and configurations of application and infrastructure, which is often challenging. The following interview extract is a response given by an interviewed practitioner from company A when asked to give an example of a DevOps practice they have had in their team:

> *The actual building of the CI pipeline has been done collaborative with ops team, who basically just provisions machines for us. They have a library of useful profiles that can be helpful, e.g., our servers have Java 8, database servers have PostgreSQL installed, etc. So, we've been pretty active in building and modifying our own CI pipeline, and we have had within ourselves actual tangible tasks to deploy our stuff to production.*

In addition to the development team interacting with the operations team to implement the deployment pipeline, the teams collaboratively work together to improve the pipeline in order to improve their processes. This is described by an interviewed practitioner when asked to give an example of a problem that the development team encountered and needed help from the infrastructure team:

> *We are using Jenkins in CI, and whenever we want to update software, we push the newest version to Jenkins and it runs unit tests. If everything goes fine, it automatically installs the new version to the test environment. At some point, we started using Selenium for automated end-to-end functional tests. We have had quite a lot of random failures with Selenium tests, for example, we did not know if the software or just the test tool was broken. Quite often, either of these were true, and sometimes I think we also run out of space in CI machines and the Jenkins build would fail because there is no space left. So we have used [name of the assigned person from Infrastructure team] to debug and have a look at what's going on when we have encountered failure that we shouldn't have.*

b) Infrastructure-as-code: This practice is central to automation dimension and entails treating infrastructure the same way developers treat code. This includes making the process of infrastructure provisioning, orchestration and application deployment reproducible and programmatic. Application code has a defined format and syntax that follows the rules of the programming language in order to be created. Typically, the code is stored in a version-management system that logs a history of code development, changes and fixes. When code is compiled (built) into an application, the expectation is that a consistent application is created. When the latter occurs, the build process is said to be repeatable and reliable. Practicing IaC means applying the same rigor of application code development to infrastructure provisioning, orchestration and deployment. All configurations should be defined in a declarative way and stored in a version management system, just like application code. With IaC, it becomes possible to quickly provision application environments and deploy application automatically and at scale as needed. The following interview extract is a response given by an interviewed practitioner from company A about how the team handles their infrastructure:

> *We use Ansible for all server automation, so we have Ansible tasks and playbooks for the infrastructure, i.e. we use Ansible projects for setting up the machines and installing. When a new machine comes, we just automatically run that and it sets up everything. Then we have other playbooks for deploying our software on top of the page.*

3) Culture: empathy, support and good working evironment for teams especially development and operations: Many of DevOps practices also involve changing culture and mindset to the one that encourages empathy, support and a good working environment for those involved in software development and delivery processes.

a) Empathy: According to majority of ML authors, development and operations need to empathize with each other and more importantly also with users of software product or service. For operations engineers, empathy helps them to understand the needs as well as to appreciate the importance of being able to push code quickly and frequently, without problems. To developers, it also allows them to recognise the problems resulting from writing code that is erroneous, unstable or insecure. Generally, empathy in DevOps culture allows software developers and operators to help each other deliver the best software possible. Information exchanges require (and can contribute to) mutual understanding. As an

 ISBN: 978-1-61208-498-5

example to this pattern includes having both developers and operations, personnel wear pagers as responsible persons to handle incidents.

b) support and a good working environment for teams: The need for mindset change and cross-discipline learning associated with DevOps necessitates the support from members within and across development and operations teams, as well as of those outside the two functions or roles. A good working environment that welcomes innovation, experimenting and stops finger pointing when mistakes are done as long as they constantly improve learning, helps to facilitate and embrace DevOps phenomenon amongst others.

4) Monitoring: Instrumenting application and aggregating monitored data into insights: This pattern involves having a continuous feedback loop that runs from the production environment to the start of the development cycle, including a complete timeline of development and operations events. It involves proactive detection and awareness of events in critical environments, such as test and production, in order to expose (know the state of) issues before they cause failures. According to practitioners, the latter is important especially for SaaS application because development teams are increasingly placing more reliance on detection and recovery than standard QA testing practices. Since the cost and time of fixing defects in production for SaaS are less than packaged software, teams tend to reduce reliance of extensive testing and instead rely more on production monitoring to detect defects as long as they can be quickly fixed. On the other hand, monitoring of advanced architectures with highly distributed systems and nodes that appear and disappear pose challenging tasks that should not only involve operations engineers. Two common patterns in monitoring include emphasis on instrumentation of application and increased use of modern tools for monitoring purposes.

a) Instrumenting application: A well-instrumented software system can provide rich data and insights about its health and performance that can be used in bug reporting, troubleshooting, feature suggestion or general finetuning of the system. Effective instrumentation of an application tries to minimize problems associated with the aggregation of large amounts of data from a variety of sources by capturing feedback as part of the application. Reliance on monitoring, developers can quickly recovery from code failures by the use of feature flags that enable or disable code functionality through configuration settings. Additionally, instrumentation can help to extend features of monitoring tools with domain knowledge. To implement this, a broad set of skills are required amongst developers that involve mainly scripting and knowledge of performance-monitoring practices, often a responsibility of operations. The following interview extract is a response given by an interviewed practitioner from company A when asked to give an example of a DevOps practice experienced by their team:

> *When we do not know how to do something, the ops team, i.e. the infrastructure team, has shared services that I use, like log aggregation services and monitoring services. We have asked them to set these up for us.*

b) Aggregation of monitoring data into insights: For comprehensive system monitoring, several tools at different system levels are used, and different tools are used to monitor application and infrastructure. Although modern tools try to provide insight into almost every aspect of system behaviour, the biggest challenge facing the tools is developing data analytics that predict problems before they become outages. Some practitioners also argue that it is impossible to catch all the issues using the out-of-the box features of such tools.

5) Measurement: different metrics are used to monitor and assess the performance of processes in development and operations activities, e.g., developers on system quality and operations on system stability. As such, instead of using proxy metrics, DevOps, according to practitioners, emphasizes the use of common metrics that are often business focused to assess and give incentive to both development and operations teams. Additionally, developers, operations and other stakeholders can use production feedback to drive decisions, improvements and changes to the systemnot just problem reports and user suggestions, but measurable data collected on how the system is working according to the conversion rate or whatever metric that the business uses to determine success.

V. Discussion

Our findings show that often practitioners vary their description of DevOps depending on whether they put their emphasis on either the goal or the means for achieving collaboration between development and operations. The most common goal according to practitioners was to reduce response time and provide fast deployment of high-quality and reliable software products and services. This finding supports the second part of the definition proposed by Penners and Dyck [7]. 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 practices attributed to it. We therefore argue that both mindset and practices be included in the definition and that the first part of the definition should not exclude the results. By doing this, we improve the definition to be *DevOps is a mind-set substantiated with a set of practices to encourage cross-functional collaboration between teams - especially development and IT operations - within a software development organization, in order to operate resilient systems and accelerate delivery of change.*

On the other hand, the DevOps phenomenon has no explicit one-size-fits-all set of practices that guide its adoption and implementation, but that common patterns can be identified from its diverse set of practices. For each pattern of DevOps, practitioners can choose different ways to implement it even though it was possible to identify similar implementation. Patterns to the DevOps practices are more useful than one-fits-all set of practices that will not suffice due to contextual limitations in organisations, e.g., team members skill and scenarios vary among companies. As an example, we can observe the reorientation of roles and teams in development and operations activities pattern in studies reported by Balalaie, Heydarnoori, and Jamshidi [19]. In the latter study, the authors depict the formation of small and autonomous teams consisting of multi-skilled persons of both development and operations activities as a DevOps practice. According to the authors, the practice helps to minimize development and operations inter-team coordination, which is important for Microservices.

 ISBN: 978-1-61208-498-5

Callanan and Spillane give empirical evidence to infrastructure and deployment automation pattern [18]. Continuous monitoring of both infrastructure and application, as a DevOps practice helping to bridge development including quality assurance and operations are described as lesson learned and experiences by authors of these studies [15] [16] [17] [18] [19] [20].

Generally, the results of our study signify the importance of considering context when describing DevOps practices. In our case study, even though most practitioners elaborated the practices in the context of cloud and web development, other contextual information, such as size and maturity, influenced the ways in which some practices are implemented. This is an important consideration that needs to be taken into account when discussing DevOps practices, i.e., to understand the context in which the practices are described.

VI. Threats to Validity

This study has a number of validity threats that need to be taken into account. We considered three categories of validity threats– construct validity, reliability, and external validity [21]– and used different countermeasures to minimize the threats. The countermeasures included: (a) consulting multiple sources in MLR (b) maintaining chain of evidence, and (c) incorporating practitioners reviews. In addition, particulary to MLR approach, three minimal standards for enhancing rigor in ML suggested by Ogawa [22] were considered.

To ensure construct validity, we only included one out of three contacted and interviewed case companies. The latter is due to not having the two other companies implement DevOps as well as the lack of DevOps understanding thereof. As a result of the selection process, our study faced some limitation with regards to the few number of interviews included in this study. This limitation serves as an opportunity for further inquiry in future works. On the otherhand, the interviews were conducted with at least two researchers to minimize researcher bias. A document describing the background and objective of the study was sent to practitioners prior to the interviews. With regards to the MLR, a threat to construct validity comes from different constraints, e.g., the selected search term as well as the inclusion and exclusion criteria of ML. To ensure construct validity and minimal rigor, prior to MLR, the objective of the study was clearly defined which served as the primary criterion for seeking and selecting ML.

To ensure the reliability of results, initial results were made available for discussion to practitioners and other researchers prior to the writing of this report. One meeting (with company representatives of the interviewed company) and a workshop with practitioners of the N4S research program were also conducted to solicit feedback from practitioners. Both events were useful for collecting feedback. We also ensured an audit trail, i.e., maintain all records and analysis in NVivo. With regards to ML, the reliability of the study would have improved further with contact and additional discussion with the authors of the ML. However, due to the large number of ML sources and diverse set of authors, at the time of writing the report, this was not feasible due to limited time.

External validity and the generalization of the findings is threatened by the small number of interviewees as described earlier as well as our reliance on ML sources as data sources for analysis. For this reason we acknowlede this limitation and , the results serve as a basis for empirical evaluations.

VII. Conclusion and Future work

The analysis of multivocal literature and interviews with software practitioners showed software practitioners use different terms and variety of practices when describing the DevOps phenomenon. We used findings from our analysis to provide more evidence that supports and builds upon the scientific definition of DevOps proposed by Penners and Dyck [6]. Our findings showed that DevOps as a phenomenon is not just a mind-set but rather some patterns of DevOps practices described by the practitioners can be identified. This study presents the identified patterns in relation to the five dimensions of DevOps, i.e. collaboration, automation, culture, monitoring and measurement. Our findings show that DevOps phenomenon is more prominent in organisations 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 a research that not only identifies the practices but also for which kinds of system, organisations and domains are DevOps practices applicable.

Acknowledgment

This work was supported by TEKES (Finnish Funding Agency for Innovation) as part of the Need for Speed project (http://www.n4s.fi/) of DIGILE (Finnish Strategic Centre for Science, Technology and Innovation in the field of ICT and digital business).

References

[1] M. Leppanen et al., "The Highways and Country Roads to Continuous Deployment," IEEE Software, vol. 32, no. 2, mar 2015, pp. 64–72.

[2] P. Rodríguez et al., "Continuous Deployment of Software Intensive Products and Services: A Systematic Mapping Study," Journal of Systems and Software, jan 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121215002812

[3] H. H. Olsson, H. Alahyari, and J. Bosch, "Climbing the "Stairway to Heaven" – A Mulitiple-Case Study Exploring Barriers in the Transition from Agile Development towards Continuous Deployment of Software," in 38th Euromicro Conference on Software Engineering and Advanced Applications. IEEE, sep 2012, pp. 392–399.

[4] T. Karvonen, L. E. Lwakatare, T. Sauvola, J. Bosch, H. H. Olsson, P. Kuvaja, and M. Oivo, "Hitting the Target: Practices for Moving Toward Innovation Experiment Systems," in 6th International Conference on Software Business. Springer International Publishing, 2015, pp. 117–131.

[5] J. Smeds, K. Nybom, and I. Porres, "DevOps: A Definition and Perceived Adoption Impediments," in 16th International Conference on Agile Software Development (XP). Springer International Publishing, 2015, pp. 166–177.

[6] N. Kerzazi and B. Adams, "Who Needs Release and DevOps Engineers, and Why?" in International Workshop on Continuous Software Evolution and Delivery. ACM Press, 2016, pp. 77–83.

[7] R. Penners and A. Dyck, "Release Engineering vs. DevOps-An Approach to Define Both Terms," Full-scale Software Engineering, 2015. [Online]. Available: https://www2.swc.rwth-aachen.de/docs/teaching/seminar2015/FsSE2015papers.pdf#page=53

[8] L. E. Lwakatare, P. Kuvaja, and M. Oivo, "Dimensions of DevOps," in 16th International Conference on Agile Software Development (XP). Springer International Publishing, 2015, pp. 212–217.

[9] J. Humble and J. Molesky, "Why enterprises must adopt DevOps to enable continuous delivery," Cutter IT Journal, vol. 24, no. 8, 2011, pp. 6–12.

[10] B. Tessem and J. Iden, "Cooperation between developers and operations in software engineering projects," in In Proceedings of the 2008 international workshop on Cooperative and human aspects of software engineering. ACM, 2008, pp. 105–108.

 ISBN: 978-1-61208-498-5

[11] J. Iden, B. Tessem, and T. Päivärinta, “Problems in the interplay of development and IT operations in system development projects: A Delphi study of Norwegian IT experts,” Information and Software Technology, vol. 53, no. 4, apr 2011, pp. 394–406.

[12] E. Tom, A. Aurum, and R. Vidgen, “An exploration of technical debt,” Journal of Systems and Software, vol. 86, no. 6, jun 2013, pp. 1498–1516.

[13] A. Dyck, R. Penners, and H. Lichter, “Towards definitions for release engineering and DevOps,” 2015.

[14] F. Erich, C. Amrit, and M. Daneva, “Cooperation between information system development and operations: a literature review,” p. Article No.69, 2014.

[15] J. Roche, “Adopting DevOps practices in quality assurance,” Communications of the ACM, 2013, pp. 1–8.

[16] D. Cukier, “DevOps patterns to scale web applications using cloud services,” in In Proceedings of the 2013 conference on Systems, programming, & applications: software for humanity. ACM, 2013, pp. 143–152.

[17] L. Bass, I. Weber, and L. Zhu, DevOps: A Software Architect’s Perspective. Addison-Wesley Professional, 2015.

[18] C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano, “DevOps,” IEEE Software, vol. 33, no. 3, may 2016, pp. 94–100.

[19] M. Callanan and A. Spillane, “DevOps: Making It Easy to Do the Right Thing,” IEEE Software, vol. 33, no. 3, may 2016, pp. 53–59.

[20] A. Balalaie, A. Heydarnoori, and P. Jamshidi, “Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture,” IEEE Software, vol. 33, no. 3, may 2016, pp. 42–52.

[21] P. Runeson and M. Höst, “Guidelines for conducting and reporting case study research in software engineering,” Empirical software engineering, vol. 14, no. 131, 2009, pp. 131–164.

[22] R. Ogawa and B. Malen, “Towards rigor in reviews of multivocal literatures: Applying the exploratory case study method,” Review of Educational Research, vol. 61, no. 3, 1991, pp. 299–305.

 ISBN: 978-1-61208-498-5