Characterizing *DevOps* by Hearing Multiple Voices

Breno B. Nicolau de França
PESC/COPPE/UFRJ
Universidade Federal do Rio de Janeiro
Rio de Janeiro, Brazil
bfranca@cos.ufrj.br

Helvio Jeronimo Junior
PESC/COPPE/UFRJ
Universidade Federal do Rio de Janeiro
Rio de Janeiro, Brazil
jeronimohjr@cos.ufrj.br

Guilherme Horta Travassos
PESC/COPPE/UFRJ
Universidade Federal do Rio de Janeiro
Rio de Janeiro, Brazil
ght@cos.ufrj.br

ABSTRACT
Recently, *DevOps* has emerged as an alternative for software organizations inserted into a dynamic market to handle daily software demands. As claimed, it intends to make the software development and operations teams to work collaboratively. However, it is hard to observe a shared understanding of *DevOps*, what potentially hinders the discussions in the literature and can confound observations when conducting empirical studies. Therefore, we performed a Multivocal Literature Review aiming at characterizing DevOps in multiple perspectives, including data sources from technical and gray literature. Grounded Theory procedures were used to rigorous analyze the collected data. It allowed us to achieve a grounded definition for *DevOps,* as well as to identify its recurrent principles, practices, required skills, potential benefits, challenges and what motivates the organizations to adopt it. Finally, we understand the *DevOps* movement has identified relevant issues in the state-of-the-practice. However, we advocate for the scientific investigations concerning the potential benefits and drawbacks as a consequence of adopting the suggested principles and practices.

CCS Concepts
• Software and its engineering → Software creation and management → Collaboration in software development**.**

Keywords
DevOps, Software Development and Operations, Multivocal Literature Review, Grounded Theory

1. INTRODUCTION
Software development organizations face the challenge of rapidly and continuously adapting themselves for unpredictable changes to achieve their business needs, in the face of an even more dynamic and competitive market [1]. In this context, many software organizations have introduced agility into their development processes to handle the frequent changes and carry out the continuous delivery of their software products and services. Such initiative contributed to reducing the development time of software releases (i.e., requirements, design, coding, and testing phases) and the periodicity between them, as well as to maintain the software constantly available for deployment.

Nevertheless, once a software release reaches its deliverable state, the deployment usually becomes responsibility of the software operations team. The reduced development time and increased periodicity between releases imply that software operations team now needs to handle a considerable amount of constant changes in the production environment, increasing its backlog. As a result, software practitioners report that such adoption of agile practices exposes inefficiencies in their release processes [2], and do not improve the operational reliability and communication between software development and operations teams. Erich *et al* [3] reinforce it by indicating that the structural and functional division of software development and operations departments has a negative impact from the practitioners' perspective.

In the face of this, it is possible to observe the recent spread of the term "DevOps" among IT professionals. This term has its origin in consultants and practitioner's conferences on software systems administration [4]. It has been discussed under diverse definitions that, in some sense, converge to the collaboration between the software development and operations teams, providing potential solutions for the scenario of constant changes and deliverables previously described. Still, IT consultants and practitioners point out many benefits associated with DevOps, such as the acceleration of the software delivery process and improvement on IT stability. However, the DevOps definition and scope remains unclear in the scientific literature or among software practitioners, with the little scientific knowledge available to support the discussion of its issues [3] [5]. Thus, we understand it is worth to investigate further the concepts concerned with DevOps, contributing to reduce the conceptual gap between the academic research and professional practices on this topic.

Motivated by these concerns and considering that DevOps is a subject characterized by the abundance of diverse types of sources but lacking scientific investigation, we conducted a Multivocal Literature Review (MLR) supported by rigorous qualitative analysis procedures (Section 2). This study characterizes DevOps (Section 3), w.r.t. its definition; issues motivating its adoption, including organizational and sociotechnical perspectives; its characteristics, such as driving principles; the universe of 51 suggested practices and the required skills for both software development and operations teams; and the potential benefits and challenges of adopting DevOps under the perspective of software engineering. Section 4 discusses related works and the relevance of the findings. Finally, we draw our conclusions based on the findings and present the road ahead in Section 5.

2. RESEARCH METHODOLOGY
2.1 Literature Review
Initially, we started by searching (*ad-hoc*) the Web for DevOps hits and trying to get a general understanding of it. This quest led us to the contextual fact that the DevOps term and discussions have their origins in industry, more precisely at conferences on

SBES '16, September 19-23, 2016, Maringá, Brazil
© 2016 ACM. ISBN 978-1-4503-4201-8/16/09$15.00
DOI: http://dx.doi.org/10.1145/2973839.2973845

system administration issues around 2009 [4].

Therefore, for this study, we considered a traditional Systematic Literature Review (SLR) [6] [7] would not be adequate, since DevOps is recent in the Industry and SLRs have limited capacity for capturing the state-of-the-practice, usually described in the gray literature. Besides, the *ad-hoc* searching suggested DevOps has little scientific investigation and evidence. However, as previously mentioned, practitioners have been discussing it on their practical perspectives. For these reasons, we considered different data sources to address DevOps. Thus, this study can be characterized as a Multivocal Literature Review [8]. The diversity of sources appears in a variety of forms (e.g., academic literature, industry reports, tool vendors' websites, and blogs), reflecting different purposes and perspectives [8]. It also comprises software practitioners, represented by movement pioneers' websites.

Even not performing an SLR, we still followed an approach consisting of planning, execution, and analysis. In the planning stage, we define the research goal as the *analysis* of different literature sources of data *with the purpose of* characterizing DevOps *with respect to* its definition, main characteristics, motivating issues, and potential benefits and challenges, *from the point of view of* SE researchers, *in the context of* ESE Group research on emerging technologies for contemporary software development. From this goal, we derived the research questions (RQs), search procedures, inclusion and exclusion criteria:

- RQ1: What is the actual meaning of the term "DevOps"?
- RQ2: What are the issues motivating the adoption of DevOps?
- RQ3: What are the main characteristics associated with DevOps?
- RQ4: What are the main expected benefits and challenges of adopting DevOps?

Regarding the searching procedures, we used structured search, i.e., a search was performed out on Google and Google Scholar. For that, we selected the following terms referring to DevOps to capture relevant data to answer the research questions: "Dev-Ops", "Dev" AND "Ops", "DevOps", "development" AND "operations". Our background allowed us to identify terms addressing particular practices, e.g., "continuous delivery" and "infrastructure as code", as being closely associated with DevOps. However, we did not include them in the search string since we are interested in the characterization under the DevOps perspective (including its related practices), rather than characterizing the practices themselves in isolation.

The following inclusion criteria supported the selection of relevant data sources: (I1) sources addressing the DevOps term or the integration of software development and operation teams; and (I2) different types of sources, such as articles, technical reports, and tool vendors' websites or blogs posts, i.e., including both academia and industry. The exclusion criteria were: (E1) repeated sources (in this case, just one was considered); (E2) duplicate sources reporting similar results (in this case, only the most complete was considered); (E3) sources published before 2009, when the term was created; (E4) material not written in English; and (E5) inaccessible sources. Particularly, the criterion E4 is justified due to the observed lack of rigor in translations from English to Portuguese on the sources such as blog posts and websites, associated with lack of their original references, which would confound the analysis.

A data extraction form was defined to capture details from the data sources including bibliographic information and relevant information for answering the RQs.

Regarding the execution stage, three researchers performed the structured search, collecting the materials by applying the inclusion and exclusion criteria in the searched hits. As one may wonder, Google and Google Scholar present millions[1] of "results" by searching for the selected terms. To handle this issue, we interleaved data collection and extraction. Thus, when we observed that no additional data could be extracted from new sources, we moved to the analysis. The extraction process was performed in pairs, still with three researchers, in such a way that a researcher extracted the relevant data and after that, another researcher reviewed that extraction. For all pieces of extracted data, at least one researcher reviewed it.

2.2 Analysis Procedures

To support the analysis process, we performed a qualitative analysis of the extracted data using procedures from the Grounded Theory (GT) [8]. Grounded Theory is a methodology for qualitative research that aims to generate theories based on the obtained data during research. However, we do not adopt GT in full as we have no ambition of generating theories in this study, but use some GT procedures just to support the analysis of relevant data and identification of concepts to answer the established RQs.

The analysis procedure is based on the coding process of analyzing (textual) data and assigning concepts to chunks of data. Such concepts represent the basic unit of analysis and are identified by breaking down the data and assigning labels to it. These labels should constantly be revisited to ensure that the conceptualization is held consistent. This coding process is handled in three different stages: *open coding*, *axial coding,* and *selective coding*. In this study, we followed the orientations presented by Strauss and Corbin [9], which assume a non-sequential order, mainly between the open and axial stages, with interleaved data collection and analysis between these stages.

Open coding is the analytical process in which researchers identify concepts, their properties, and dimensions in the data. In this process, the data are fragmented and conceptually labeled in *codes*. In turn, *codes* may represent actions, events, properties, and interactions that are relevant for the researchers. During the *open coding* process, the concepts should constantly be compared with each other to find similarities, and then they should be grouped to form categories. Therefore, in *axial coding*, categories are associated to their subcategories. It is also constantly done as new categories emerge. Finally, in *selective coding*, all categories are unified around the central core category and relationships are established among these categories.

To answer RQ1, we performed *open*, *axial* and *selective coding* to understand the concepts and their relationships better to define DevOps around a core category (Development and Operations Altogether in Section 3.2). For the other RQs, only the *open* and *axial coding* was carried out, since the goal was to categorize the aspects that would support their answering.

Aiming to support the coding process, we adopted the QDA Miner Tool[2], in which all extracted data was imported. The top right of Figure 1 exemplifies the open coding process. In this case, we assign concepts to pieces of extracted text (highlighted), which represent DevOps characteristics. For each new code, we compare it to the existing ones, aiming to understand whether it refers to the same concept.

[1] Searching "DevOps" on Google returns about 15.300.000 hits.

[2] http://provalisresearch.com

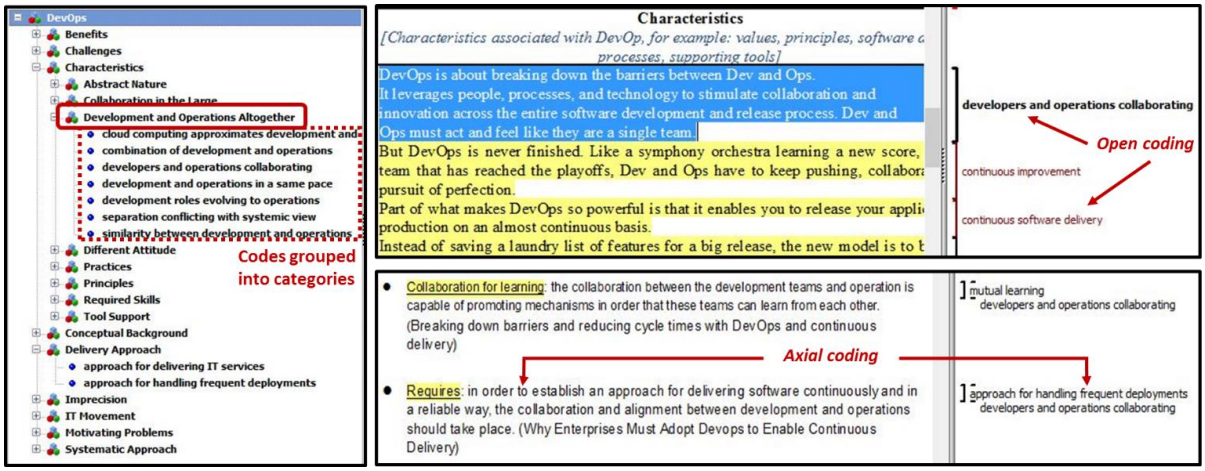

Figure 1. Codes grouped into categories (open coding) and relationship among different categories (axial code)

After that, we established relationships among the codes through reasoning about the descriptions regarding DevOps definitions to generate the categories (axial coding).

Thus, the name assigned to a particular category aims at representing, at a higher abstraction level, all concepts inherent to the codes about it. For instance, the left side of Figure 1 exemplifies a category (*Development and Operations Altogether*) created from the relationship among some codes (*vertical or hierarchical relationship*). Besides, we established relationships between codes of different categories (*horizontal relationship*) that enabled us to understand better the categories needing additional details. The bottom right of Figure 1 shows a relationship among codes from different categories. For instance, the relationship "Requires", referring to codes: "*approach for handling frequent deployments"* and "*development and operations collaborating"*, which belong to different categories.

3. RESULTS

3.1 General Numbers from the Review

During the structured search, we selected 50 data sources based on the application of inclusion and exclusion criteria. After reading and extracting all data, we reduced to 43 sources as no further useful data could be extracted. These sources loosely mention DevOps but do not discuss it on a consistent basis for our research questions. Table 1 presents the sources included by type.

Table 1. Selected sources

Types	#Sources	References
Academic Journal	3	[10] [11] [12]
Academic Tech. Report	2	[13] [14]
Book	1	[15]
Website	12	[16] [17] [4] [18] [19] [20] [21] [22] [23] [24] [25] [26]
Academic Conference Papers	8	[27] [28] [29] [30] [3] [31] [32] [33]
Industrial Journals	8	[34] [35] [36] [37] [38] [39] [40] [41]
Industrial Tech. Report	8	[42] [43] [44] [2] [45] [46] [47] [48]
Proceedings Preface	1	[49]
Total	43	

More than 69% (30) of the sources are gray literature, represented by websites, books, industry journals, and industry technical reports. Therefore, this distribution reinforces the idea that DevOps is little explored by the academic community yet.

3.2 DevOps Definition (RQ1)

In our literature review, we observed that several distinct definitions were assigned to the DevOps term. Figure 2 presents the main concepts defining DevOps according to our dataset, identified in the stage of *open coding*. Both practitioners and researchers presented these definitions, having no consensus among them. Besides, we could observe data sources explicitly mentioning DevOps having no precise definition.

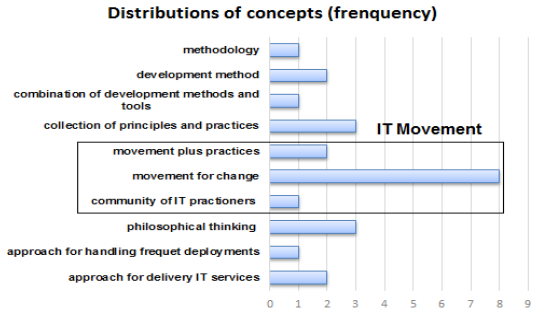

Figure 2. Concepts defining DevOps

Erich *et al.* [3] also could not identify a consolidated definition for DevOps through a systematic mapping study. Even so, they defined DevOps as a framework, but emphasizing the problem regarding the lack of consensus among the reviewed studies. However, considering it as a framework would require clear instantiation steps or rules, which is not available for DevOps.

As we identified different concepts regarding the DevOps definition, we verified their actual meaning against three dictionaries[3]. For instance, we discarded terms defining DevOps as a methodology [46] or a method [47] since these terms imply on the existence of a systematic approach for performing DevOps or introducing its practices into organizations. Such strategy

[3] Oxford, Cambridge, and Merriam-Webster.

played an essential role in understanding what DevOps mean. Therefore, aiming a better understanding, we identified the relationships among categories (Figure 3) like the ones in Figure 2. These categories were established in the *axial coding*.

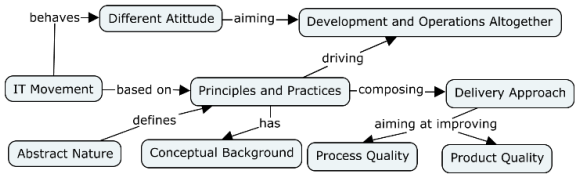

Figure 3. Inter-related concepts (categories) composing the DevOps definition

This way, we identified that DevOps is more frequently associated with the idea of a movement of Information and Communication Technology (ICT) practitioners (Figure 2). Hence, we emphasize that the meaning of term "*movement*" is, according to the adopted dictionaries, associated with a group of people working together to achieve a common goal, as well as a situation in which people change their way of thinking or doing something.

As a result of the coding process applied to the data, we synthesized that *DevOps is a neologism representing a movement of ICT professionals addressing a different attitude regarding software delivery through the collaboration between software systems development and operation functions, based on a set of principles and practices, such as culture, automation, measurement and sharing*. Such collaboration aims at improving the quality of development and operations processes, as well as the quality of software products and services.

3.3 Motivating Issues (RQ2)

We organized the issues motivating software organizations to introduce DevOps into the following categories.

Organizational Structure and Policies: refer to the problems faced by organizations regarding the impact of their organizational structure and policies on their IT performance. The data sources pointed out that the software organizations have been isolating functionally and physically the software development and operations teams [18] [37] [32]. Hence, organizations separate software projects from their operation, and consequently software maintenance is hampered since the team responsible for the development of software is not the same one that will maintain it after the first release [30]. In this context, the software development team is measured by the number of delivered features while the software operations team by the systems stability. However, using system stability as a measure of the success of operations triggers an immediate response of preventing deployments as much as possible to avoid unstable releases. In turn, it will clash with the productivity measurement for development, focused on delivered features [37] [36] [46] [43]. All the issues associated with the team's structural division increases the development cycle time, delaying delivery of valuable functionality or corrections, reducing collaboration, and increasing frustration and lacking of trust among teams. Therefore, a systematic approach is required to improve the whole organization, focusing on collaborative actions.

External Pressure: the current scenario for software development requires organizations to respond faster due to market conditions, such as business changes, customers frequently demanding new features in a reliable way, and the development and support regarding different devices [29] [33]. Moreover, the growing demand for software as service has also led these organizations to restructure their business processes, and now they become responsible for both the software development and operation [34].

Release Process: considering the frequent demand for new features, the agility in software development processes affects the release process, leading to bottlenecks in the operations processes [39] [41] [44]. Such phenomenon exposes issues between software development and operations teams, in which the release process is unable to deliver new changes, hindering the continuous delivery of value to customers. For instance, the lack of parallelism between the development and deployment activities within releases increases the lead-time and slows down opportunities of early problems discovering [46]. Another issue regarding the release process is the fear of changing the system after it reaches stability, making organizations adopt rigid and bureaucratic processes for managing releases, which requires more effort and time to introduce new changes or corrections [46].

Quality Demands: Currently, the software systems are becoming increasingly sophisticated (large and tightly coupled) [35], demanding even more non-functional features, such as maintainability, interoperability, scalability [28], and a high degree of reliability [37]. The difficulties in meeting these requirements are associated with characteristics of both the software systems and their development processes: long lifecycles, which makes harder to carry out changes; and the deployment and testing processes, which require a significant amount of time and effort for changes and rework. Another issue is concerned with the lack of developers' knowledge on how to handle non-functional features and involved tradeoffs. Also, to compensate this lack of knowledge, software operations teams use more powerful IT infrastructures, though more expensive and complex [37].

Sociotechnical Issues: issues associated with both social and technical aspects, such as cultural differences between development and operations negatively affecting communication [30], the need for teams with new skill sets [10], and the intensification of existing problems due to geographic distribution of teams, leading to lack of trust among them [40]. Finally, the ineffective communication among different stakeholders, including development and operations teams, is pointed out as one of the main issues motivating the adoption of DevOps [38] [39] [46] [29] [49] [30] [43]. Part of the communication issues are credited to long release cycles hampering the continuous project progress monitoring and product quality [37]. In addition, it reduces the amount of feedback from stakeholders regarding the current product being built [28]. Consequently, such ineffective communication leads to software features delivering marginal value for the business and defects discovered late in the lifecycle.

3.4 Characterization (RQ3)

We observed a set of principles, skills, and tools to support the practices characterizing DevOps. Also, DevOps is an abstract concept that requires being instantiated to specific organizational contexts, instead of applying a specific set of technologies.

3.4.1 Principles

Figure 4 shows the principles associated with DevOps grouped into six categories, which are briefly explained in the following.

Social Aspects: despite all technical principles, many of the DevOps characteristics are associated with social aspects among the software development and operations teams. For instance, the so-called DevOps culture recognizes trust as a relevant

characteristic for influencing organizational change [41] [42] [30].

Automation: claimed to be one of the core principles of DevOps due the benefits it could promote. It considers that manual, and repetitive tasks can be automated to reduce unnecessary effort and improve software delivery [17] [45]. Hence, automation would improve not only the delivery speed, but also the infrastructure consistency, productivity of teams, and repeatability of tasks.

Quality Assurance: to assure the quality of both development and operations processes as products. This principle supports the implementation of DevOps practices since it links different stakeholders (development, operations, support, and customers) to perform activities in an efficient and reliable way, as well as the product and services meeting established quality standards [30].

Leanness: some DevOps practices are based on Lean Thinking principles [50] (Figure 4). DevOps requires a lean process as it intends to ensure a continuous flow to develop and deliver software regularly, in small and incremental changes [35] [36] [46] [43]. Therefore, fostering constant and fast feedback between the development and operations, as well as with customers [35].

Sharing: information and knowledge are disseminated among individuals to promote the exchange of personal learning and project information. In this sense, individuals should spread relevant information, for instance, those regarding how to implement and to perform practices recommended in the context of DevOps [17] [20]. Besides, information regarding changes or new characteristics of project or product should be spread to the involved individuals (software development and operation teams), and it promotes the workflow visibility [37].

Measurement: an important principle often instantiated by collecting efficient metrics to support the decision-making in the software development and operations lifecycle [17] [37] [38].

Many data sources explicitly pointed out four DevOps principles, which are associated with the acronym CAMS: Culture, Automation, Measurement and Sharing [37] [30] [4] [27] [25]. From our observations, culture in CAMS is part of what we capture as Social Aspects. Furthermore, we could identify two additional principles: Quality Assurance and Leanness. Although, the sources do not explicitly mention Leanness as a core principle, we identified many of the recommended practices are implementations of principles from the Lean Thinking.

3.4.2 Practices

Although the principles reveal the fundamental ideas and values characterizing DevOps, the practices materialize those principles into daily development and operations activities. Thus, one can recognize the adoption of DevOps regarding maturity through the observation of running practices into the organizations.

Initially, we observed several recommended practices in the context of DevOps. Hence, to understand how and when such practices are performed, we identified commonalities and grouped them into categories and subcategories (axial coding). Figure 5 groups the 51 practices identified into three broad categories: Common (33), Development (9) and Operations (9). Besides, it shows the codes (actual practices) belonging to each category and the associated number of sources they appear and occurrences considering the whole data. Due to space limitation, we discuss only most mentioned practices.

Common practices refer to those performed by both software development and operations teams. It includes the subcategories: Collaborative (18), Procedural (14) and Services (1).

Collaborative practices aim to foster collaboration among teams and their members. Role rotation is a recurrent practice [36] [49] [20] [2] [45]. It allows a team member to understand problems and solutions better from other teams, exchanging experience and collaborating to solve common problems. Another recurrent practice is to notify the right responsible people regarding issues on the software lifecycle allows coordinating the product team by triggering actions and giving awareness so that one can anticipate or react to problems [47] [19] [17] [31] [21].

Procedural practices represent a set of practices performed by software development and operations teams together and concerned with their coordination. We observed continuous software delivery [35] [12] [43] [21] [14] and deployment pipeline [37] [46] [14] as the most frequent practices.

It may represent an indication of their relevance in the context of DevOps. Continuous software delivery follows the idea of extending continuous integration cycles to include also software release. For that, both developers and operations are required. Also, the continuous delivery practice abstracts a process from checking the source code until the software release into production. Hence, the deployment pipeline is one practice to implement continuous delivery in an automated way.

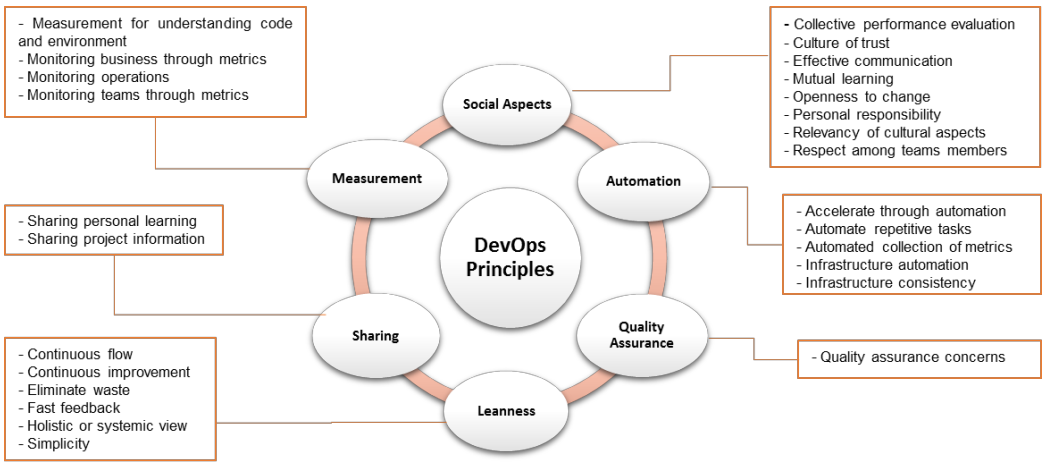

Figure 4. DevOps Principles

Concerning the Services subcategory, it represents practices performed together by the software development and operations teams and concerned with the use of different service models (Infrastructure as a Service - IaaS, Platform as a Service - PaaS, and Software as a Service - SaaS). We observed data sources pointing out that automation, mainly of operation tasks, could be supported by cloud services providing platforms (PaaS) and infrastructure (IaaS) facilities. Additionally, development and operations can be interfaced by exposing their tasks as services, for instance: monitoring, version control, deployment, and others.

Among the development practices, Continuous Integration is an agile practice inherited in DevOps [32] [16] [21] [45], in which developers routinely merge, multiple times per day, their code into a version control repository. Hence, each change triggers an automatic set of tasks involving build, tests, and deployment. Therefore, it develops integration and testing practices in a reliable way, ensuring the build works in a specific environment.

Finally, we grouped into the Operations category the practices performed mainly by the system administrators. Infrastructure configuration management and automated infrastructure provisioning are recurrent practices. In large-scale environments, operations staff should maintain a configuration management strategy to determine the right infrastructure configuration for a given software version. It helps to keep track of and to automate changes in the environment, avoiding unnecessary effort and rework. Concerning the automated infrastructure provisioning, this practice aims to instantiate development, testing, and production environments in an automated way so they can be consistent. The implementation of this practice is accomplished through the intensive use of tools or service models.

Most of the identified practices are performed by developers and systems administrators together. Also, such practices aim to (1) support and encourage cooperation between development and operation software areas, (2) support the coordination of activities performed by these areas, and (3) integrate the execution of these activities as services. Finally, we identified two sets of practices: one specific to development and other specific to operations.

3.4.3 Required Skills

An issue also driving organizations to adopt DevOps is the need to organize teams with a set of competencies meeting the current software development demands. Therefore, the set of individuals' skills is an important characteristic in the context of DevOps.

In this sense, members of the software development and operation teams should be able to expand their abilities beyond their specific

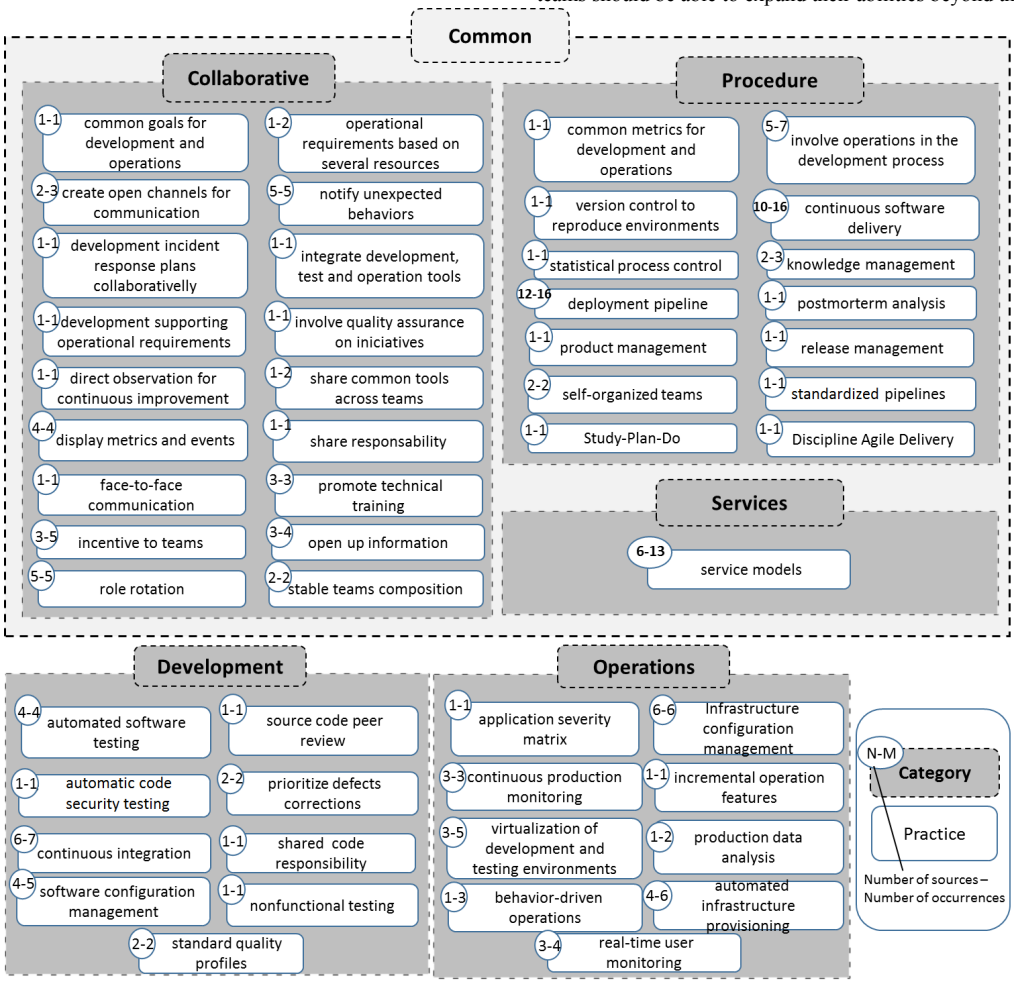

Figure 5. DevOps Practices

roles to understand and act on specific tasks. It means that DevOps requires team members with multiple skills (social and technical) [14] [45] and cross-functional teams [18] [43] [20] [2]. These characteristics potentially blend expertise from multiple sources to solve an issue or to explore potential solutions.

We identified coding as a required skill for both development and operation [2]. Coding or scripting is necessary due to the need of automation of tasks, including infrastructure provisioning and deployment. Also, it is essential to have people with experience in measurement as it can contribute to risks analysis and mitigation [14]. We could also identify the need for team members with math skills. Foundational skills in math improve understanding of measurement, monitoring and performance analysis of team members, resources, and software systems [45].

A critical failure factor is an ineffective communication among stakeholders, justifying communication as a relevant skill in the DevOps context since it promotes the collaboration and integration between development and operations teams [18].

3.5 Benefits and Challenges (RQ4)

The data sources "claim" a myriad of advantages regarding the adoption of DevOps, which we identified as potential benefits as no evidence of their achievement is given. Moreover, we grouped them into organizational, processes, software product and people.

DevOps is claimed to be an enabler to the achievement of organizational goals efficiently. For instance, practitioners report DevOps practices as a competitive advantage, potentially enabling the increase of organizational IT performance, and consequently, the profit, market share, and productivity also increase [45]. Thus, high-performance organizations tend to work closer with IT to plan initiatives, avoiding sudden and unreasonable demands over IT teams. Besides, the use of automated solutions, such as configuration management and infrastructure provisioning, would lead to organizational benefits as they offer consistency of environments across different stages of software lifecycle [47] [38]. Ideally, it reduces efforts and costs, including the time needed to increase the machinery and to recovery it from losses [13]. Other potential benefits are: increase in the number of customers using the company's software/services [48]; improve operational efficacy and efficiency of the organization as changes are based on operations and users feedback [49] [30] [22]; development and operations answering to business needs instead of their own interests, possibly reducing business risks [37] [29] [2]; and the focus on new features, due to automation [48].

Development and operations processes benefit from the constant integration and testing, along with automated infrastructure provisioning, are practices that enable to track and solve problems in processes and/or activities [43] [45]. In general, many principles and practices are reported as a cost saving alternative [47] [37] [41] [45], since they enable a more efficient use of resources (human and infrastructure), reducing waste in the delivery process [10] [43] [31]. Besides, we observed the sources claiming these practices reduce cycle and lead time of development and operation activities in an end-to-end perspective [37] [39] [11] [2] and, consequently, improve time-to-market [35] [41] [46] [48] [2] [14]. Moreover, such practices and principles are claimed to: improve process and product visibility [2] [38]; support the identification and reduction of processes variability to improve predictability [41]; make possible the deployment of applications with no outage through automated mechanisms [28] [37] [17] [2]; and provide a flexible process to accommodate change requests at any time in the lifecycle [34].

Regarding benefits associated with the software product, the combination of DevOps practices such as code sharing, continuous integration, test-driven techniques and automated deploys would expose problems earlier in the lifecycle [23], enabling anticipation or prevention of defects [32] [30] [43]. In this scenario, problems associated with merging and integrating would be reduced, since changes are smaller and rapidly integrated [41]. Such practices aim at achieving working software constantly by adopting incremental development [43] [37]. We also identified benefits as supporting the improvement of system reliability, outstanding collaboration efforts from development and operations [28] [12], scalability and durability for software systems [28]. DevOps practices aim to meet the expectations of users, delivering software with high quality [34].

People-related benefits refer to the achievements of members of the development and operation teams, as well as to customers. One claimed benefit is the communication and collaboration between the software development and operations teams [38] [29] [14], promoted by their practices and principles that aim to break down functional and physical separation among such teams. Thus, in this collaborative atmosphere, teams are encouraged and feel technically qualified to work in different areas as a single unit [18]. We found sources indicating that development teams improve their productivity over time when practices such as continuous integration and constant feedback are adopted [45]. DevOps practices are also claimed to improve the understanding of a system under development [41], since small teams are joined by larger ones in a cross-functional structure. Moreover, we identified that DevOps aims to increase the satisfaction of organization's employees [18], and customers [16].

Beyond the benefits, there are also challenges associated with the culture, management, technical aspects and IT infrastructure.

Culture: organizations usually resist to do any changes on the way things work [38], particularly when its culture is involved [36]. It may also be the case of partner and customer organizations [39]. This way, extra effort is always required to overcome it and convincing through metrics and achievements that such change is a benefit. This challenge becomes even harder when the teams lack integration and organizational incentive [15]. In organizations where top management does not work collaboratively with operational staff, software development and operations teams have difficulties in self-organizing and making decisions. Besides, it may be associated with the lack of a clear understanding of what DevOps means [21] [2], which hampers people to focus on specific aspects such as processes and tools.

Management: the implementation of DevOps requires considering all the impacted teams [36], understanding development and operations roles and responsibilities [48]. Therefore, the alignment of strategies and processes for software development and operations teams aimed at achieving a common goal is a challenge [29]. In this sense, the involvement of top management seems to be essential to drive and support the DevOps initiatives [48] [2], as in initiatives consisting of considerable organizational change. Automation supports many of the practices recommended in the context of DevOps. However, the use of automation tools requires proper management and high investment [39] [49]. Moreover, the adoption of DevOps is hampered when processes are managed in a rigid form within the organization, allowing low or no level of flexibility [36] [31].

Infrastructure: DevOps aims at constantly and rapidly delivering valuable software, requiring automation to support activities such as build, deploy to production-like environments, testing at

different levels and so on. However, one of the challenges for the implementation of DevOps is the lack of adequate infrastructure to support the automation of these tasks [2]. Moreover, not having infrastructure provisioning in an automated procedure represents an obstacle. The ability to keep operations stable is also identified as a challenge hindering the adoption of DevOps [31] [45].

Technical: A DevOps core characteristic is the flexible working processes. Therefore, it is challenging to introduce principles when the working processes need to comply with strict industry standards and regulations [37] [48] [39]. As DevOps aims to automate every repetitive activity, including those related to operations [49], the need for coding skills by the system administrators to develop and maintain the infrastructure provisioning scripts represents a challenge. Also, systems with tightly coupled architectures also represent a technical issue [41]. Systems with this characteristic are harder to maintain, demanding additional effort from development and operation teams.

4. DISCUSSION

4.1 Related Work

Erich *et al.* [3] performed a systematic mapping study aiming at understanding the influence of the relation between development and operations on Information System development. Based on that, they define DevOps as a conceptual framework for reintegrating development and operations based on the so-called "main concepts" related to DevOps, identifying then culture, automation, measurement, sharing, services, quality assurance, structures, and standards. Additionally, they also identified issues associated with the relationship between software development and operations teams, also how DevOps can alleviate such issues.

Similarly, Lwakatare *et al.* [5] also performed a literature review supplemented by interviews with practitioners. From that, they identified four general dimensions of DevOps: collaboration, automation, measurement, and monitoring. Moreover, they identified issues addressed by DevOps, such as poor communication and manual tasks for operation processes, as well as expected outcomes from its adoption.

Both works highlight the need for more research on DevOps, which is a contribution of our review. The captured concepts and dimensions are strongly related to the principles we categorized in section 3.4.1, which presents a greater and more structured set of concepts obtained through conceptualization and categorization.

4.2 Relevance of Findings

In general, our findings are based on the qualitative analysis of several data sources having different levels of rigor, including gray literature that is mostly opinion and experience based (anecdotal evidence). It means that we have almost no scientific evidence on it. However, we argue the need for such variety by considering the characterization goal, the research questions, the origin of the DevOps movement on practitioners' conferences, and the amount of available knowledge that practitioners provide on this topic. Besides, among the sources, we have blogs [4] [19] and white papers [37] from the movement pioneers that created the term DevOps and should be considered. Then, considering the amount of 69% of the data as gray literature, we adopted GT as a systematic and rigorous methodology to acquire a deep understanding of what they mean by DevOps and to synthesize the different perspectives on a critical view.

Regarding the actual findings, we do not advocate the identified principles and practices compose the definitive set suggested in the DevOps movement. It may be incomplete or even includes other principles and practices. As DevOps has no clear scope regarding the set of recommended practices [21], we understand these practices are reasonable for the context of this research. Still, many principles and practices are credited to agile and lean software development. It applies in particular to development practices (Section 3.4.2). Operations practices seem to compose the state-of-the-practice of systems administrators. Besides, the common practices, in which development and operations teams work collaboratively, appears to be mostly speculative w.r.t. their effectiveness and potential benefits. It is also important to comment that no evidence was found regarding the achievement of potential benefits, including the scientific papers we analyzed.

Although the lack of scientific evidence, we understand that our findings compose an important starting point for further scientific investigation on these practices in the context of such collaboration between software development and operations teams in software intensive organizations. Furthermore, we highlight that both technical and social aspects are concerns of these organizations, and the motivating issues are still challenging them on how to succeed in their current development scenario.

As mentioned in comments by [18] and [46], the practices in the context of DevOps are not new when taken in isolation, and there are few organizations adopting subsets of these practices for years. Some ones have more success than others. The claim is that such discussion helps to disseminate both development and operations practices that practitioners believe to be effective [49] and the need for a culture of collaboration between these functional areas. We believe that one possible novelty may be, as it is in agile software development, the specific blend of principles and practices to foster collaboration between development and operations, rather than a new methodology or set of tools. However, the original blend depends on each organizational environment, culture, domains, and other factors.

Finally, despite the technical issues, the social challenges are mostly concerned with the cultural change of an organization, which is already identified as a critical factor in many other research areas like software process improvement and information management systems. Thus, any movement towards introducing DevOps recommended practices should consider it.

4.3 Threats to validity

The threats to validity are discussed according to [6] and [51].

The theoretical validity concerned with searching and selection bias was handled by possibly capturing multiple sources and applying the inclusion criteria (Section 2.1), which were previously established. Even not performing a comprehensive search, we searched in open search engines and snowballed backward to reach relevant data sources. One limitation relies on not considering videos as information sources in the protocol.

Publication bias is also a concern in multivocal literature reviews, where collected data is not exclusively peer reviewed and accepted/published materials. That is true, even apart from the peer-reviewed works, as tool vendors and posts from blogs of the DevOps pioneers and enthusiasts have biased opinions regarding potential benefits of the adoption of recommended practices. We mitigated it by critically analyzing each term from the referred concepts and terms, comparing against all collected information through the constant comparison method and recurring to dictionaries when no conceptual reference could be obtained.

Regarding descriptive validity, our extraction form and process were unbiasedly defined before the execution to answer the RQs strictly. Also, at least one researcher reviewed all information.

The results are internally generalizable as we achieve theoretical saturation on the identification or appearance of new codes and categories. However, from the external perspective, we cannot argue that as our review aimed an initial characterization.

Finally, interpretive validity is achieved when the conclusions are drawn reasonable given the data. A threat in interpreting the data is researcher bias. For the analysis, we adopted rigorous GT procedures in executed parallel by two researchers to avoid bias and solve inconsistencies. Then, the codebooks from each researcher were compared iteratively against each other, remaining only consensual information on the resulting codebook.

5. CONCLUSIONS

The SE community should react to movements stemmed from Industry by systematically understanding them to ground the R&D on the topic. In this sense, we provide a characterization of what practitioners and academics discuss as DevOps. The analysis of the different sources of information allowed us to answer reasonably the RQs, which we discuss in the following.

RQ1: What is the actual meaning of the term "DevOps"? We have indications that we reach a sound definition of DevOps. First, we acknowledged that some research has been conducted on this topic, but without a clear definition of what DevOps mean [3] [5]. That is one contribution that regards to the stated definition, reached through the solid methodology for qualitative synthesis, namely GT. Second, we included in the study sources naturally concerned with DevOps and originated from the same context it was created, i.e., the industry, with a strong focus on systems administration and development consultancy.

RQ2: What are the issues motivating the adoption of DevOps? As far as we observed, DevOps is associated with a myriad of problems that foster its adoption in software organizations. Initially, the market and stakeholders exert an external pressure for continuously changing software systems and increasing the number of quality demands. From an internal perspective, the organizational structures and policies, as well as their manual and bureaucratic release processes hamper the timely answer to these constant demands. Besides, sociotechnical aspects influence initiatives to overcome the issues, aggravating the situation.

RQ3: What are the main characteristics associated with DevOps? From an abstract perspective, we analyzed DevOps characteristics regarding principles, practices, and required skills (Section 3.4). It is important to remind that these categories emerged from the data, i.e., they were not established before the analysis. Principles address concerns with social aspects like culture, automation, measurement, sharing, quality assurance, and leanness. Besides, practices are the way one can observe how DevOps manifests into an organization. Several practices were identified, but we concentrate on the common practices for development and operations, which may regard collaboration, procedural, and services issues. Finally, we identified the claim for an extended skill set required to perform DevOps practices.

RQ4: What are the main expected benefits and challenges of adopting DevOps? We could not observe evidence on the actual measurement or perception of expected benefits and challenges. However, we do identify claimed benefits addressing organizational, process, people, and product perspectives by adopting DevOps principles and practices. Also, challenges arise along the way, as expected in any organizational change.

From this study, we can assert that DevOps is a movement of IT practitioners, which has identified a set of important issues on the field and that the research community should investigate approaches to handling them. Most of the suggested principles and practices need further investigation w.r.t. their effectiveness against the motivating issues. Moreover, we understand that additional investigation on characterizing DevOps may not bring value to the study of the practices, tools, and methodologies to put these things together in different organizational contexts.

6. ACKNOWLEDGMENTS

We acknowledge the support of Daniel Karam Venceslau during data collection, CNPq and CAPES for the research grants.

7. REFERENCES

[1] Boehm, Barry. Making a Difference in the Software Century. *Computer*, 41, 3 (March 2008), 32-38.

[2] PUPPET LABS & IT REVOLUTIONS PRESS. *State of DevOps Report*. Puppet Labs, 2013.

[3] Erich, Floris, Chintan, Amrit, and Maya, Daneva. A Mapping Study on Cooperation between Information System Development and Operations. In *Proceedings of the 15th PROFES* (Helsinki, Finland December, 2014), 277-280.

[4] Mueller, Ernest. What Is DevOps? *The Agile Admin*. 2010. Retrieved from: http://goo.gl/mD3LU1.

[5] Lwakatare, Lucy Ellen, Kuvaja, Pasi, and Oivo, Markku. Dimensions of DevOps. (Finland 2015), Springer.

[6] Biochini, Jorge, Mian, Paula Gomes, Natali, Ana Candida Cruz , and Travassos, Guilherme Horta. *Systematic Review in Software Engineering*. COPPE/UFRJ, Rio de Janeiro, 2005.

[7] Kitchenham, Barbara A. and Charters, S. *Guidelines for Performing Systematic Literature Reviews in Software Engineering*. Keele University, 2007.

[8] Ogawa, Rodney T and Malen, Betty. Towards Rigor in Reviews of Multivocal Literatures: Applying the Exploratory Case Study Method. *Review of Educational Research*, 61, 3 (1991), 265-286.

[9] Strauss, Anselm and Corbin , Juliet. *Basics of qualitative research: Procedures and techniques for developing grounded theory*. SAGE, Newbury Park, 1990.

[10] Roche, J. Adopting DevOps Practices in Quality Assurance. *Communications of the ACM*, 56, 11 (2013), 38-43.

[11] Liu, Yuhong, Li, Chengbo, and Liu, Wei. Integrated Solution for Timely Delivery of Customer Change Requests: A Case Study of Using DevOps Approach. *International Journal of U-& E-Service, Science & Technology*, 7, 2 (2014), 41-50.

[12] Mohamed, Samer I. DevOps shifting software engineering strategy Value based perspective. *IOSR Journal of Computer Engineering (IOSR-JCE)*, 17, 2 (April 2015), 51-57.

[13] Economou, Frossie, Hoblitt, Joshua C., and Norris, Pat. *Your data is your dogfood DevOps in the astronomical observatory*. 2014.

[14] Erich, Floris, Amrit, Chintan, and Daneva, Maya. *Report: DevOps Literature Review*. University of Twente, 2014.

[15] Walls, Mandi. Building a DevOps Culture (2013).

[16] Berczuk, Steve. Agile Teams Care About DevOps. 2011. Retrieved from: http://goo.gl/Rw2pWZ.

[17] Hedemark, Magnus. DevOps in Straight English – Enter the Buzzword. *RedHat Developers*. 2014. Retrieved from: http://goo.gl/W5LMws.

[18] Nelson-Smith, Stephen. What Is This Devops Thing, Anyway? *Jedi (Patrick Debois' Blog)*. 2010. Retrieved from: http://goo.gl/qsmNqO.

[19] Debois, Patrick. Devops Areas - Codifying devops practices. *Jedi (Patrick Debois' Blog)*. 2014. Retrieved from: http://goo.gl/25sCB3.

[20] Humble, Jez. There's No Such Thing as a "Devops Team". *Continuous Delivery*. 2012. Retrieved from: http://goo.gl/9bTKhb.

[21] Tesar, David. DevOps Practices. *ITPROGUY.com*. 2015. Retrieved from: http://goo.gl/iKPgTJ.

[22] Geer, David. Internap's DevOps Culture: PrivateStack + CD = ? [Read On & Draw Your Own Conclusions]. *DevOps.com*. 2015. Retrieved from: http://goo.gl/vVAqqZ.

[23] NEW RELIC. DevOps. *New Relic*. 2015. Retrieved from: http://goo.gl/IDfddk.

[24] Garnichaud, Neil. What Exactly is DevOps. *DrDroobs*. 2012. Retrieved from: http://goo.gl/V2ev4J.

[25] Willis, John. What DevOps Means to Me. *Chef.io*. 2010. Retrieved from: https://goo.gl/stl6Go.

[26] Corriere, Chris. The devOpsSec Dilemma: Effective Strategies for Social Networking. *DevOps.com*. 2015. Retrieved from: http://goo.gl/fIp7wL.

[27] Bang, Soon K, Chung, Sam, Choh, Young, and Dupuis, Marc. A Grounded Theory Analysis of Modern Web Applications - Knowledge, Skills, and Abilities for DevOps. In *RIIT'13* (2013), ACM.

[28] Cukier, Daniel. DevOps patterns to scale web applications using cloud services. In *Proceedings of the 2013 conference on Systems, programming, & applications: software for humanity* (2013), ACM, 143-152.

[29] Hussaini, Syed W. Strengthening harmonization of Development (Dev) and Operations (Ops) silos in IT environment through Systems approach. In *IEEE 17th Int. Conf. Intelligent Transportation Systems* (2014), 178-183.

[30] Erich, Floris, Amrit, Chintan, and Daneva, Maya. Cooperation between information system development and operations: a literature review. In *Proceedings of the 8th ACM/IEEE ESEM* (Torino, Italy September 18-19, 2014).

[31] Kim, J., Meirosu, C., Papafili, I., Steinert, R., Sharma, S., Westphal, F. J., and Manzalini, A. Service provider DevOps for large scale modern network services. In *Int. Symp. on Integrated Network Management, IFIP/IEEE* (2015).

[32] Waller, Jan, Ehmke, Nils C., and Hasselbring, Wilhelm. Including Performance Benchmarks into Continuous Integration to Enable DevOps. (2015), ACM SIGSOFT Software Engineering Notes, 1-4.

[33] Pengxiang, Ji and Leong, Peter. Teaching Work-ready Cloud Computing Using the DevOps Approach. In *Int. Symp. on Advances in Technology Education* (Singapore 2014), Nanyang Polytechnic.

[34] Sussna, Jeff. Cloud and DevOps: A Marriage Made in Heaven. *Introducing DevOps to the Traditional Enterprise / eMag*, 14 (June 2014).

[35] Phillips, Andrew. Preparing for Continuous Delivery in the Enterprise. *Introducing DevOps to the Traditional Enterprise / eMag*, 14 (June 2014).

[36] Manglani, Kamal and Bothello, Gerald. DevOps - Pivoting Beyond Pockets. *Introducing DevOps to the Traditional Enterprise / eMag*, 14 (2014).

[37] Humble, Jez and Molesky, Joanne. Why Enterprises Must Adopt Devops to Enable Continuous Delivery. *The Journal of Information Technology Management*, 24, 8 (2011).

[38] Shamow, Eric. Devops at Advance Internet: How We Got in the Door. *The Journal of Information Technology Management*, 24, 8 (August 2011).

[39] Fitzpatrick, Lawrence and Dillon, Michael. The Business Case for Devops: A Five-Year Retrospective. *The Journal of Information Technology Management*, 24 (August 2011).

[40] Phifer, Bill. Next-Generation Process Integration: CMMI and ITIL Do Devops. *Cutter IT Journal*, 24, 8 (2011).

[41] DeGrandis, Dominica. Devops: So You Say You Want a Revolution? *The Journal of Information Technology Management*, 24, 8 (August 2011).

[42] Smith, David M. *Hype Cycle for Cloud Computing*. Gartner Research, Inc, 2011.

[43] Duvall, Paul. Breaking down barriers and reducing cycle times with DevOps and continuous delivery. *New Relic*. 2012. Retrieved from: http://goo.gl/dqhwod.

[44] Azoff, Michael. *DevOps Advances in release Management and Automation*. Ovum, 2011.

[45] PUPPET LABS & IT REVOLUTIONS PRESS. *State of DevOps Report*. Puppet Labs, 2014.

[46] Michelsen, John. Dysfunction Junction: A Pragmatic Guide to Getting Started with DevOps. *CA Technologies*. 2013. Retrieved from: http://goo.gl/A1ZAYM.

[47] APPDYNAMICS, INC. From Dev to Ops: An Introduction. 2014. Retrieved from: https://goo.gl/qn1xUi.

[48] CA TECHNOLOGIES. *TechInsights Report: What Smart Businesses Know About DevOps*. CA Technologies, 2013.

[49] Limoncelli, Thomas A. and Hughes, Doug. DevOps: New Challenges, Proven Values. *;login:*, 36, 4 (2011), 46-48.

[50] Womack, James P and Jones, Daniel T. *Lean thinking: banish waste and create wealth in your corporation*. Simon and Schuster, 2010.

[51] Petersen, Kai, Vakkalanka, Sairam, and Kuzniarz, Ludwik. Guidelines for conducting systematic mapping studies in software engineering: An update. *IST*, 64 (2015), 1-18.