Lappeenranta University of Technology
School of Business and Management
Degree Program in Computer Science

Mohamed Khaled Aljundi

Tools and Practices to Enhance DevOps Core Values

Examiners: Professor Ajantha Dahanayke
MSc Dmitrii Savchenko

Supervisor: Professor Ajantha Dahanayke

ABSTRACT

Lappeenranta University of Technology
School of Business and Management
Degree Program in Computer Science

Mohamed Khaled Aljundi

Tools and Practices to Enhance DevOps Core Values

Master's Thesis

78 pages, 10 figures, 16 tables, 4 appendix

Examiners: Professor Ajantha Dahanayke
MSc Dmitrii Savchenko

Keywords: DevOps, CAMS, Automation, Culture, Measurement, Sharing.

The modern development companies are facing numerous challenges to keep up with the endless technological and business requirements of the market to develop software systems and maintain them. DevOps introduced as a set of tools, rules, and practices in order to create more efficient ways to deal with challenges related to software development and maintaining its state during development and production. CAMS is DevOps's core values, and it represents aspects related to the software development environment like culture, automation, measurement and sharing. However, implementing DevOps culture is tricky and contains obstacles. The thesis uses design science methodology to develop a model that combines practices and tools that enhance DevOps core values.

ACKNOWLEDGEMENTS

This work wouldn't see the light without the endless support of my supervisor Prof. Ajantha, my special gratitude and thanks to her for guiding me through the different phases of the thesis. On the other hand, the technological insights of Mr. Timo from Eficode (Finland) had helped to refine this work and makes reach its current state. Last but not least, special thanks to Mr. Araya and Mr. Escalante from Costa Rica, their suggestions and expertise gave me new perspectives in regard to the topic of the thesis.

TABLE OF CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

CAMS	Short for Culture, Automation, Measurement and Sharing
CC	Code Checking
CD	Continuous Delivery
CI	Continuous Integration
CM	Configuration Management
Dev	Development team
DevOps	Dev: stands for development. Ops: stands for operations
DSM	Design Science Methodology
DSR	Software as a Service
Ops	Operations team
SLR	Systematic Literature Review
VIR	Virtualization

1 INTRODUCTION

1.1 Background

The need to overcome the latest challenges in the software development industry has driven the IT community to refine, polish and improve existing software development methodologies (agile, scrum...etc.) to suit the rapid changes in software development industry [1] [2]. Better product quality, testing efficiency, shorten time to market, customer satisfaction, stable and reliable releases and more factors have become the essential requirements of any modern software development organization to gain a stable position in the market [2] [3] .

The agile method focuses on how the dev team copes with increasing project changes without compromising other aspects affected by the project momentum. With such environment where changes are coming, it is essential to distribute information among the project stakeholders like customers, project managers, quality assurance, testers, and developers. Whenever new changes are submitted, it is important to update the related artifacts, repositories to avoid code defects and effort consumption due to wrong or outdated information [1]. A movement addressed these challenges to improve the processes to keep up with advancement in the software industry, this movement is called the DevOps. Each source and literature define DevOps differently, but many sources use a definition stated by Chef company; “Cultural and professional movement, focused on how we build and operate high-velocity organizations, born from the experiences of its practitioners - DevOps definition based on Chef” [4]. Moreover, Hussaini [5] defines the term DevOps as development and operations of information technology and it was rising to respond to the acknowledgment of the gap that exists between development team and operation team within an organization, this gap comprises the 4cs (communication, cooperation, culture, and collaboration) [6].

DevOps solves agile methodologies drawbacks and provide a set of tools and practices to ensure a smooth development life cycle even after releasing the product [6] [7]. However, one of the reasons for implementing DevOps, is to remove silos among different IT departments and preventing them from acting as individual entities within an organization

[6] [5]. Breaking silos between teams allow them to share and to collaborate with least conflicts to create reliable and stable workflows between IT operations and development departments. Achieving such allows a quicker production releases and ease problems detection in the present and the future [2] [5]. However, achieving the desired level of transparency requires organizational and cultural changes along with refining practices and frameworks [6]. Some organizations face another set of challenges such as failing to select the right set of tools, since dev team heavily depend on these tools during continuous integration and continuous deployment, thus, choosing the wrong tools might affect the efficiency of the work. Moreover, low trust in automation processes which is considered as one of DevOps core values, makes it difficult to efficiently implement continuous integration and continuous deployment [8] [9]. Additional challenge is required when organizations try implementing different aspects of modern software development life cycle such as version control, configuration management, automation, and continuous integration [10]. Adoption of DevOps practices is not easy and might not be smooth due to the changes needed to stress on the core of the followed methodologies [11].

This thesis aims to improve the DevOps practices to help organizations enhancing DevOps core values while implementing DevOps. Describing DevOps practices and tools benefit other researchers and organizations alike by addressing and identifying the current state of DevOps in the industry and academia, which in turn helps to reduce time consumption and bypass potential obstacles. The main outcome of this thesis is to address the practices and tools to adopt DevOps and enhance its core values by creating a model.

1.2 Aims and Objectives

This study aims to address the best practices and tools required to enhance the core values by identifying the list of most common practices in the field from the literature and industry point of view. And create a model that include the findings and help to enhance DevOps core values. Those objectives are as follows:-

- Identification of the Practices followed to enhance DevOps core values which are reported in the literature.
- Addressing the tools used to enhance DevOps core values which are reported in the literature.

- Identification of the tools and practices followed to enhance DevOps core values which are being used in the industry.
- To propose a model to enhance culture, automation, measurement and sharing DevOps values.

1.3 Research Questions

Table 1. Research Questions

Research Questions	Research Method	Expected Outcome
RQ1.What are the practices and rules followed to enhance DevOps core values CAMS reported in the literature?	Systematic Literature Review	**EO1**: A list that represent practices to enhance DevOps core values reported in the literature
RQ2.What are the tools used in order to enhance DevOps core values CAMS reported in the literature?	Systematic Literature Review	**EO2**:A list that represent tools to enhance DevOps core values reported in the literature
RQ3.What are the practices and tools used in order to enhance interdepartmental communication to improve DevOps core values CAMS reported in the industry?	Interview	**EO3**: validate and compare the outcomes of EO1 and EO2 and create a list
RQ4. How to enhance DevOps core values?	Artifact creation	**EO4:** The artifacts based on the results from EO1, EO2 and EO3.

2 RESEARCH METHODS AND STRUCTURE

2.1 Research Methods

Organizations use information systems for raising the effectiveness and efficiency of knowledge sharing processes along with developing and communicating the knowledge "knowledge concerning both the management of information technology and the use of information technology for managerial and organizational purposes" [12]. Thus acquiring the knowledge is essentially important for research activities surrounding information systems in general. There are two ways to obtain the knowledge, behavioral science, and design science. Behavioral science uses natural science research methods to gain the required knowledge. The main aim of behavioral science is to develop, support and utilize theories that explain phenomena circling the design, implementation, supervision, and practice of information systems. On the other hand, design-science research (DSR) method has its origins from sciences and engineering. Essentially Design science is a problem-solving approach with the aim of creating innovative solutions. Innovations might include ideas, practices, models, and new technical capacities through which the use of information systems can be effectively and efficiently accomplished [12].

The thesis aims to create a model, which solves certain problem. There are many research methods that help to achieve that. However, design science methodology proven to be supportive in researches that require problem solving through building artifacts (models, algorithms…etc.). The iterative approach used in design science methodology helps to refine and improve the model the researcher aims to accomplish.

Alan Hevner describes DSR as "It is fundamentally a problem solving paradigm. It seeks to create innovations that define the ideas, practices" [12]. DSR is solution-oriented knowledge based on scientific variables used to draw an artifact to solve a problem or improve existing solution (check figure 1). DSR helps to bring insights to complex domains. Additionally, DSR aimed to design artifacts to solve issues in a complex domains in which requires innovation to make an improvement while creating solutions based on knowledge and then execute the solution to check its validity [12].

DSR became popular in the past years due to the flexibility it provides while innovating new ideas to solve particular problem or improve a current solution. On the contrary, other research methods or paradigms may not have a significant impacts on the thesis due to the proposed domain problem and the scope it covers. However, DSR approach provides flexible yet standardized process which shows the researcher the roadmap to create, improve, validate, refine and finally communicating the proposed solution (model). There is no "right" way to conducts DSR, but Alen Hevner in his paper "DESIGN SCIENCE IN INFORMATION SYSTEMS RESEARCH" provides guidelines which assist the research to draw a map on how to conduct DSR. These guidelines can be modified and altered based on each domain.

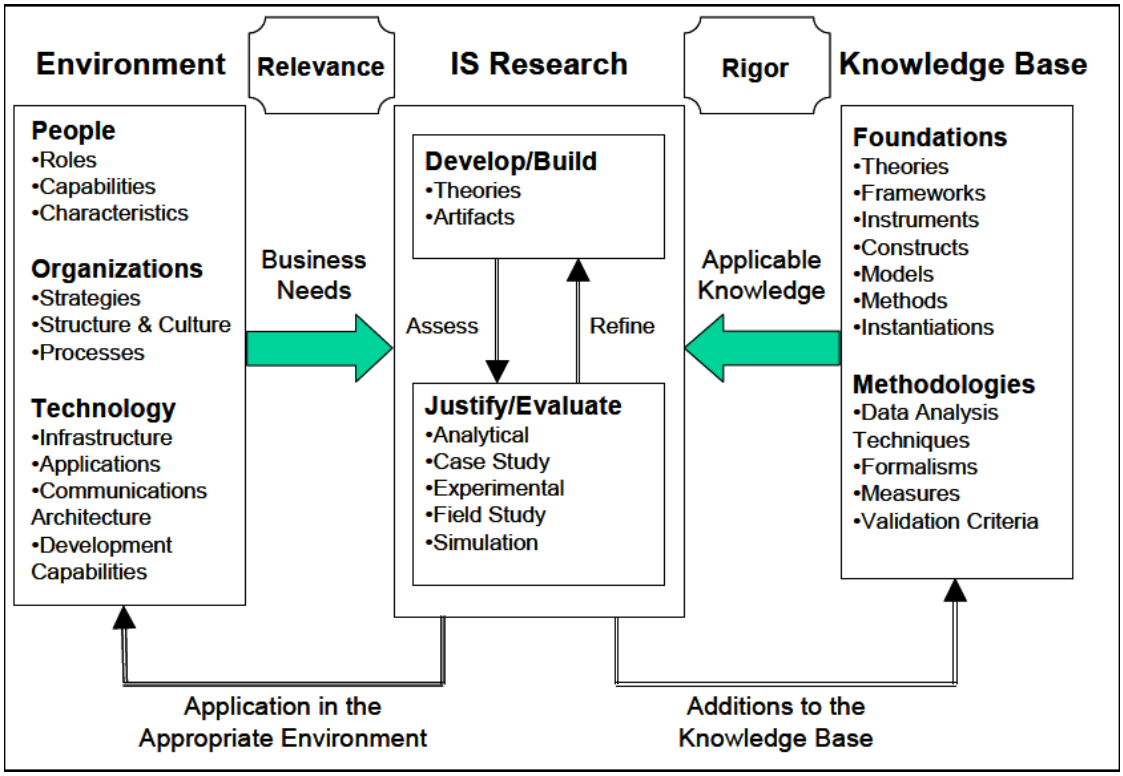

Figure 1. DSR framework. [12]

2.2 Problem Relevance

To understand the scope of the problem, the thesis started with a literature review to understand the background of DevOps. The literature review assisted the researcher by providing knowledge about the current state of DevOps. Overcoming DevOps challenges require an organizational and cultural change along with refining practices and framework

[6]. However, some organizations face another set of challenges such as failing to select the right set of tools, since dev team depends on these tools during continuous integration and continuous deployment, thus choosing the wrong tools might affect the efficiency of the work. Low trust in automation processes which considered one of DevOps core, makes it difficult to implement continuous integration and continuous deployment [8] [9]. Organizations face numerous challenges implementing different aspects of modern software development life cycle such as version control, configuration management, automation, and continuous integration [10]. These difficulties and obstacles have motivated the researcher to conduct this thesis to ease adopting DevOps processes. The next table represent the relationship between research process and research techniques in DSR.

Table 2. Relationship between research process and research techniques in DSR

Research Process	Research Techniques	Research Objectives (refer to table 1)
Problem Relevance	1- Literature Review	To address the problem and understand the background of the project
DSR evaluation	1- Systematic literature review 2- Interviews	Achieve RQ1 Achieve RQ2 Achieve RQ3 Achieve RQ4
Validation	Case studies	To validate RQ1, RQ2, RQ3 and RQ4

The aim of this thesis is to develop a model comprises DevOps tools and practices to assess organizations to enhance DevOps core values. Design science guides the researcher while creating a model by developing knowledge through a set of guidelines to achieve knowledge and understand the problem domain based on fixed artifacts which should be developed in earlier stages. The results of the systematic literature review and the interviews are used in DSR to design a model and then to test it.

2.3 DSR Evaluation

"It must be described effectively, enabling its implementation and application in an appropriate domain." [12] [13] Alan Hevner about artifacts. This artifact could be a method, model, constructs or instantiations. However, the desired artifact should be based on viable

knowledge [14]. The researcher uses DSR for the aims of creating a model which represent a set of tools and practices to support and enhance DevOps core values. The model is structured into five main parts culture, automation, measurement, sharing, and tools. The findings and the knowledge extracted from SLR and interviews are used to create the model. Each part of the model represents context related knowledge sorted in a particular classification. The model requires knowledge to build it, so the researcher conducts a SLR and interviews to give the sources the needed knowledge to generate the model.

2.3.1 Systematic Literature Review

"A systematic literature review is a means of evaluating and interpreting all available research relevant to a particular research question, topic area, or phenomenon of interest. Systematic reviews aim to present a fair evaluation of a research topic by using a trustworthy, rigorous, and auditable methodology" [15]. There are different reasons to perform SLR, one of which is to present a framework/background to position new research activities [16] which are what this thesis aims to achieve. This thesis uses SLR to answer RQ1 and RQ2 to define the set of tools, rules, and strategies reported in the literature to enhance interdepartmental communication to improve DevOps core values CAMS.

2.3.2 Interview

Interviews are the second research technique in this thesis, the researcher is intending to use it to check to which extent the literature is accurate regarding the tools and practices used to enhance DevOps CAMS values and filling any gaps occurred while conducting SLR, and to help to create a model according to design science paradigm. The researcher shows the list of practices and tools used to enhance DevOps CAMS values to the interviewees to create a model.

The interviews can be conducted in any way suitable for the interviewees since all the interviewees are professionals from the industry. The researcher gives the interviewees the flexibility to decide the suitable style of communication for them, whether it is a face-to-face interview, online, email... etc. The researcher focuses on in-depth questions which directly related to the core of the topic which is how to enhance DevOps core values CAMS. The results help to identify and test any gaps between the results collected from SLR.

Additionally, the researcher keeps records of the interviews with the interviewees for further revision and validation if required.

Proposed Questions:

1. To which extent do you think DevOps is being implemented in your organization?
2. How did your organization break the interdepartmental silos? Especially between Dev and Ops teams?
3. How does your organization implement culture and sharing values of DevOps?
4. Culture and Sharing are important values of DevOps, what are the tools used in your organization in order to enhance them?(what are the tools that facilitate those values)
5. To which extent is automation is being used, and what processes have been automated?
6. What are the tools used in automation?
7. Does your organization implement measurement practices?
8. What exactly is being measured?
9. What are the impacts of those measurements?
10. What are the practices followed in your organization to implement measurements?
11. What are the tools used in measurements?

2.4 DSR Validation

After conducting SLR and interviews, the researcher has the data to build the desired solution by following DSR. The main aim of conducting DSR is to generate a structured model to contribute and assess organizations to enhance DevOps core values. Another aim is to combine existing DevOps tools and practices reported in the literature and the ones which are implemented in real life. At this stage, the model is measured and observed in the context of how well the model supports the proposed solution. The evaluation process of the artifacts may include:

1- Functionality comparison (to make sure the objectives are met)

2- Feedback from expert.

3- Case Studies

At the final stage, it is necessary to present the artifact and its impact on the solution, the research shall present the artifact design, evaluation results and the model effectiveness in the discussion part followed by a conclusion. The researcher is looking for an innovative solution to solve the problem of enhancing DevOps, DSR is the most suitable technique to achieve the desired solution.

2.5 Research Strategy

Figure 2 address the main steps followed to implement DSR:

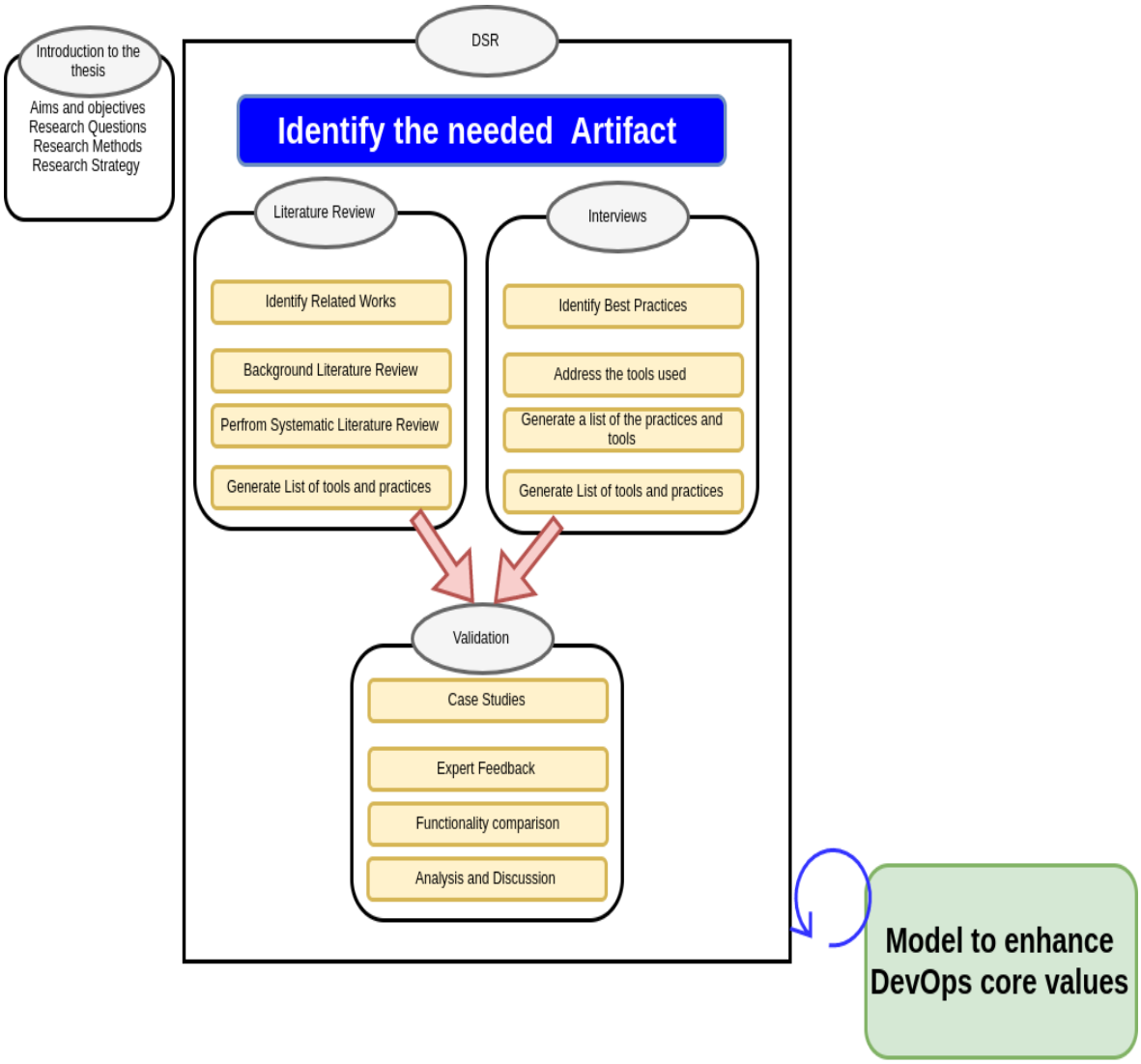

Figure 2. DSR research strategy for the thesis.

3 SYSTEMATIC LITERATURE REVIEW

3.1 Related Work

DevOps is a new movement, the related literature resources are limited and have different focuses and scopes. However, there are several papers that are important for the research. Syed W Hussaini [5] published a paper on strengthening harmonization of Dev and Ops teams. He argues that implementing DevOps was a difficult process due to the difference in perspectives between different teams. Based on the study he conducted, the conceptual definition of particular terms can differ from one team to another. An aim of a team might conflict with an aim of the other team. The paper aims to produce a model which helps to identify the key common objectives among dev and ops teams, focusing on critical success factors, identify the most critical challenges and understands how to engage and develop DevOps strategies [5].

The authors Cois, C. A., Yankel, J., & Connell, A [1], address the communication challenge in their paper "Modern DevOps: Optimizing Software development Through Effective System Interaction". The authors believe that modern software development life-cycle requires an accurate recording of data and makes it available for sharing between the different teams involved in the process of developing software products. Their paper introduces a new model that include a designed autonomous system actors and processes. In the paper they mentioned that the aim is to create "A generalized model of DevOps will be presented and analyzed, offering a formalization of the communications and actors requisite to any effective software development process" [1].

Erich, F., Amrit, C., & Daneva, M. [7] have conducted a systematic mapping about DevOps. Their focus was on the main characterization and DevOps features and concepts such as DevOps culture, automation, measurement, sharing services and quality assurance. In their paper, they have explored the most relevant texts in the literature related to DevOps standards, practices, relevant characteristics and supporting factors. The report is interesting for the thesis, since it summarizes the best practices followed, which helps to analyze different DevOps characters from different perspectives related to communication and management.

Yiran, Z., & Yilei, L. (2017) [8] discuss the Challenges and Mitigation Strategies of Using DevOps during Software Development. Yiran and Yilei have conducted a systematic literature review about the challenges of DevOps, and they listed and classified the challenges into different classifications with a detailed explanation about each challenge.

3.2 DevOps Background

3.2.1 Naming History

In 2008 a software developer called Andrew Shafer started a session placed at the Agile Conference in Toronto, the title of the lecture was "Agile infrastructure ". Only one person attended the session, Patrick Debois. Later Patrick and Andrew met and discussed, they have agreed to form the "Agile Systems Administration Group". Later in June 2009, a conference took place at O'Reilly Velocity called O'Reilly Velocity 09 conference. Patrick Debois watched the conference and tweeted on his Twitter account he could not attend. One organizer tweeted back by advising him to organize his own Velocity event in Belgium. Four months later, Patrick Debois organized the event, but he needed a name for his event so he broke the first three letters of the words development and operations, and adding the word "days" so the full name became DevOpsDays. The event started with large attendees of developers, tool smiths, administrators and others. After the event, Patrick Debois created a hashtag on Twitter called DevOps, this hashtag became popular and allowed the coining of the term DevOps. [17]

3.2.2 The need for DevOps

The regular processes of software development have put a strict border between the business department, development department, and operations department and all the other groups involved in the process of delivering a product. This way is inefficient due to silos it creates between groups involved in the development life cycle [6] [5]. But it wasn't a big issue a few years ago because the products were being updated, upgraded and replaced every once and while (quarterly, yearly) [18]. However, right now software products are being updated multiple times per day, if any mistake or a bug found it might affect the brand image of the product, and the users always expect new releases and fixes on a regular basis. The software development methods faced rapid challenges. From there, a movement that started in 2009 lead to the birth of DevOps to provide a better quality product even after the official release.

Before DevOps, Agile approach came in to solve the issues faced through an iterative approach to handle the feedback and fix the bugs [6] [19]. The modern development life cycle requires more phases to ensure the stability of the product in values to tackle the new challenges and offer approaches that help companies to produce the market such as continuous development and deployment to keep improving the product and release new versions of it [19].

DevOps aims to break down the silos between inter departments and allowing everyone involved in the process of delivering a product such as system engineers, DBs and network engineers, security to communicate and share their concerns. DevOps is not only a set of tools and procedures to follow, it is a culture for IT departments [20]. For example, Google calendar helps to set up meetings on the calendar to remember to come on time. But the calendar can't force anyone to come to the meetings, it is up to the individuals to do it or not, but the calendar can help to facilitate the process to remind the individuals of the meeting. DevOps is like the calendar example, it can provide the tools necessary to get the job done, but the cultural aspect is the most important one. On the other hand, measurement is an important aspect of DevOps, and it is important to improve the product because if a product is measurable then it is possible to improve it [20]. Measurement is the ops responsibility (servers, traffic, storage... etc), but if DevOps culture is being implemented, the developers can make sure that applications export useful data that the ops team can use and benefit from. The dev team can develop services while bearing in mind the necessity to integrate more features into the systems like recording and analysing data. Other aspects can be measured that are not particularly important to dev and ops groups such as customer activity, transactions per second, measuring errors and failure [21].

3.2.3 What DevOps provide

Efficiency and effectiveness drive modern software engineering. Fast and High-level collaboration is a key to solve rapid complex day-to-day tasks. Teams used to use agile methodologies to ensure adaptation to day-to-day challenges, changes, requirements, and customer needs while maintaining projects. The key to maintaining such an operational environment is effective communication between stakeholders to synchronize the continuously changed information (repositories, artifacts, etc.) and makes it available to the targeted stakeholders. Communication problems (or incorrect information) result in a set of

negative impacts upon the whole development lifecycle. There are massive challenges to store, document and record communication. There is a certain information that requires special treatment such as testing, tasking, source code, system configuration, and monitor and update the project status in real time to avoid defect injection. Many movements came to life with the aim of solving the challenges related to modern software engineering requirements [1].

DevOps is one of the movements that emerged to bridge the gap between software operations and software developers and ensure harmonious working environment and flexible communication. Interdepartmental communication is one of the many issues DevOps is trying to solve. DevOps uses the famous manufacturing concept "pipelines" by combining DevOps core values CAMS. DevOps continuous delivery pipeline makes continuous delivery of software reality by allowing the operations and developers to continuously updating, fixing, upgrading and adding software features faster and more efficient than existing methods. Furthermore, due to the structural environment DevOps sustains, the operational staff are responding quicker to the requirements changes, testers are also able to produce an automated testing cycles[22][23][24].

3.2.4 DevOps Core Values CAMS

John Willis mentioned "After the first US based Devopsdays in Mountainview 2010 Damon Edwards and I coined the acronym CAMS, which stands for Culture, Automation, Measurement and Sharing" [25] [26]. Like any movement, DevOps had defined the core targeted values that impact the modern requirement of software development lifecycle, the acronym that describes those values is CAMS stands for Culture, Automation, Measurement, and Sharing.

Culture: DevOps aim to remove the silos between teams and enhance interdepartmental communication. Organizations and individuals are consuming a lot of time due to the traditional way of interdepartmental communication (handoff documentation, queues), and in some organizations, it might be difficult to ask other people for help outside the official channels. DevOps enhance a healthy communication to allow all teams to communicate and eliminate the communication obstacles. Encouraging a communicative culture is a key to having a productive environment [27].

Automation, Perhaps the most visible aspect of DevOps. Many people focus on the productivity gains (output per worker per hour) as the main reason to adopt DevOps. Not only automation used to save time, but also to prevent defects, create consistency, and enable self-service [28].

Measurement, DevOps supports continuous delivery and deployment. Continuous delivery requires continuous improvement of the product, thus, DevOps supports measurement and makes it one of its core values because the movement believes it is difficult to improve without the ability to measure. Decisions based on visible and easy-to-read data are the key to having the right choices. The data should be available, transparent, accessible, related visually [21].

Sharing, DevOps realized the power of sharing and the significant impact it brings. Within the organization sharing tools, findings, defects, and experiences enable the individuals who share similar interests and needs to meet. Such a move makes the process of finding new opportunities to collaboration much easier, eliminating duplicated work along with having a powerful sense of commitment. DevOps also, promote sharing resources (code, documentation… etc.) outside the organization with the related community. It helps the organization to find new features, find defects and energize the individuals [29].

3.3 Systematic Literature Review

"A systematic literature review is a means of evaluating and interpreting all available research relevant to a particular research question, topic area, or phenomenon of interest. Systematic reviews aim to present a fair evaluation of a research topic by using a trustworthy, rigorous, and auditable methodology," [15]. This research pursues the research by using a Systematic literature review (check figure 3) because the theses try to answer a question which requires a high-level evidence checking criteria. SLR offers the needed techniques to minimize bias to produce reliable findings that lead to efficient decision making.
The systematic literature review follows the steps proposed by Kitchenham and Charters [16]. The steps are:

1. **The search questions**

2. **The search process**
3. **Study selection**
4. **Quality assessment**
5. **Data extraction process**
6. **Data extraction results**
7. **Discussion of research questions**
8. **Study limitation**

After developing the research questions, aims, and objectives a review protocol is created to select the related studies. This protocol is important for data extraction. On the other hand, the initial search keywords are “DevOps”, “interdepartmental”, “CAMS” (as full or separated words). After that, the searches for resources based on the specified searching and protocol criteria for the studies selection are carried out. The process of data extraction takes place after finishing the practical screening and quality appraisal. Finally, combines the collected facts and start the analysis to prepare for the final review.

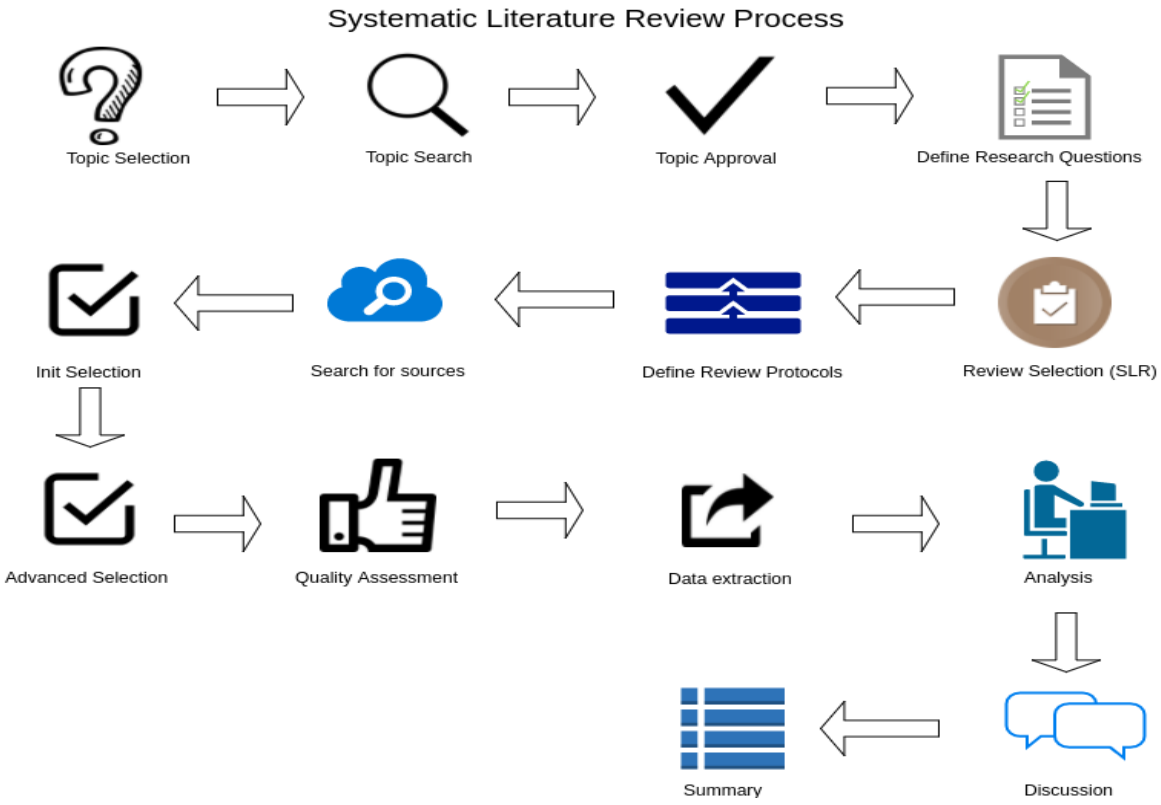

Figure 3. SLR strategy in the thesis

3.3.1 SLR Processes

“A systematic literature review (often referred to as a systematic review) is a means of identifying, evaluating and interpreting all available research relevant to a particular research question, or topic area, or phenomenon of interest. Individual studies contributing to a systematic review called primary studies; a systematic review is secondary study," [16]. By conducting Systematic literature review aims to identify all the possible published literature resources related to the proposed topic and then test and analyze the relevant literature to the proposed topic.

3.3.1.1 Review protocol

In any research, there is a possibility of researcher bias, the analysis could be derived from the researcher expectation. Defining a review protocol before conducting the research is necessary to reduce the bias [16]. The review protocols for this thesis are:

- **Search Strategy**: In this strategy, designs the pattern on how to search for primary studies, and what are the terms used. And the researcher should describe the resources required including the targeted databases along with types of the texts used (E Libraries, journals, conference proceedings...etc.)
- **Study selection criteria:** selection criteria are important because it helps to determine if the studies are included or excluded.
- **Study selection procedures:** It is a protocol which describes how the selection criteria is implemented
- **Study quality assessment:** It creates a quality criteria to make the assessment process more accurate among the individual studies.
- **Data extraction strategy:** It addresses the main information required from the primary studies and how it is going to be obtained. The protocol should specify an appropriate validation process if any data manipulation or assumptions required.
- **Synthesis of the extracted data:** It describes the synthesis strategy by clarifying if a formal meta-analysis is planned or expected and if so how the techniques are used [30].

3.3.1.2 Search Strategy

RQ1 focus on finding the practices and rules to enhance interdepartmental communication to improve DevOps core values CAMS. From RQ1, the main search focus particular

keywords such as "**DevOps**", "**rules**", "**interdepartmental**", "**enhance**","**CAMS**" and "communication". The research keywords can have synonyms words. Table 3 and Table 4 list the search keywords for RQ1 and RQ2 and their synonyms.

RQ2.focus the tools used to enhance interdepartmental communication to improve DevOps core values CAMS reported in the literature.

RQ1 table:

Table 3. RQ1 keywords search string.

W1	W2	W3	W4	W5	W6
1-DevOps	1-Rules	1-Interdepartmental	1-enhance	1-CAMS	1-Communication
2-Development and operations	2-Practices	2-Depatments	2-improve	2-Culture	
	3-methods			3-Automation	
				4-Measurement	
				5-Sharing	
				5-values	

The searching string along with the Boolean operators AND and OR are as follow:

- (W1-1 OR W1-2) AND (W2-1 OR W2-2 OR W2-3) AND (W3-1 OR W3-2) AND (W4-1 OR W4-2) AND (*W5) AND (W6-1 OR W6-2).
- (W1-1 OR W1-2) AND (W2-1 OR W2-2 OR W2-3) AND (W3-1 OR W3-2)
- (W1-1 OR W1-2) AND (W3-1 OR W3-2) AND (*W5) AND (W6-1 OR W6-2).
- (W1-1 OR W1-2) AND (*W5).
- (W1-1 OR W1-2) ND (W2-1 OR W2-2 OR W2-3)
- (W1-1 OR W1-2) AND(W2-1 OR W2-2 OR W2-3) AND (*W5)
- (W1-1 OR W1-2) AND(W4-1 OR W4-2) AND (*W5)
- (W1-1 OR W1-2) AND (*W5)
- (W1-1 OR W1-2)

Table 4. RQ2 keywords search string.

W1	W2	W3	W4	W5	W6
1-DevOps	Tools	1-Interdepartmental	1-enhance	1-CAMS	1-Communication
2-Development and operations		2-Depatments	2-improve	2-Culture	2-Contact
				3-Automation	
				4-Measurement	
				5-Sharing	

				5-values	

The searching string along with the Boolean operators AND and OR are as follow:

- (W1-1 OR W1-2) AND (W2) AND (W3-1 OR W3-2) AND (W4-1 OR W4-2) AND (*W5) AND (W6-1 OR W6-2).
- (W1-1 OR W1-2) AND (W2) AND (W3-1 OR W3-2)
- (W1-1 OR W1-2) AND (W2) AND (W3-1 OR W3-2)
- (W1-1 OR W1-2) AND(W2) AND (*W5)

The resources should be related to “software engineering” or “computer science” tags from different databases and online libraries. The literature issuing data shouldn’t be older than 2009. The research libraries that are going be used to search for the literature are:

1. IEEE
2. ACM Digital Library
3. Wiley online library
4. Emerald
5. Proquest
6. Sciencedirect
7. Springer

3.3.1.3 Study selection criteria

“Study selection criteria are used to determine which studies are included in, or excluded from, a systematic review. It is usually helpful to pilot the selection criteria on a subset of primary studies” [30]

Selection criteria for this thesis uses three categories, any text that doesn't meet any level of the criteria is excluded from the selection given in the table 5 below:

Table 5. Level of relevance.

Level of relevancy	Criteria
Low level relevancy:	a. The full text exist and in English b. Contains search words in the title and abstract
Medium level relevancy	c. Has relevant information in the introduction and conclusion which is related to DevOps
High level relevancy	d. The text has direct information related to the proposed research questions

3.3.1.4 Study selection procedures and criteria

It is a protocol which describes how the selection criteria are implemented, the selection procedures and criteria of this thesis focus on the texts published after 2009 because DevOps is a new concept, and any text from before 2009 does not hold a value to DevOps. Other related criteria are title relevancy, abstract relevance and if there is any duplication from different databases.

3.3.1.5 Data extraction strategy

It addresses the main information required from the primary studies and how it is going to be gained. The protocol should specify an appropriate validation process if any data manipulation or assumptions required. For the thesis the information required to extract the data to make it more useful for the later analyses are as follows in Table 6:

Table 6. Required information for data extraction.

Text information
Title
Writer
Publishing year
Reference
Search keywords
Abstract
The context of the text
Level of relevance
Relation to RQ1
Relation to RQ2

3.3.1.6 Study quality assessment

The quality criteria helps the assessment process, to ensure the accuracy among the individual studies. Table 7 demonstrate the used quality attributes.

Table 7. Quality assessment table.

Assessment Quality Question
Research aim is well described
RQ1 or RQ1 mentioned or discussed in the Text
The literature Review is clearly structured
DevOps is clearly the core of the discussion

3.3.1.7 Synthesis of the extracted data

Table 8 describes the synthesis strategy by clarifying if a formal meta-analysis is planned or expected and if so how the techniques are being used [30].

Table 8. Synthesis strategy table.

practices, tools	Reference No	Author	Related to RQ1	Related to RQ2

3.4 Conducting SLR

Based on the keywords and strings search defined in the previous section, conducted the search in seven different databases. Excluded articles which are not related to information technology, computer science, and software engineering to ensure the relevancy of the papers. However, since DevOps is a new concept, and the thesis does not have a wide range of related papers, used the selection criteria mentioned in the previous section to sort the papers before preparing them for extraction.

To ensure selecting high-quality papers which might contribute to the thesis, a set of keywords defined to search for the metadata of the papers at first and then conducted a full-text search. To summarize the results of the search, the articles which contribute to the thesis are few. However, the researcher has to extract the data from the articles to draw the desired conclusion. After performing the search, the researcher has excluded duplicate resources and the topics which are not related to one of the following fields: computer science, information technology, and software engineering.

Table 9. Number of selected papers.

No	Database Name	Articles Found	Init selection
1	ACM	109	82
2	Wiley Online L	58	12
3	Emerald	9	4
4	IEEE	143	52
5	Proquest	54	7
6	Science Direct	67	5
7	Springer	33	0

Table 9 conclude the total selected papers after the initial selection are 162 paper. Those articles have explicitly mentioned one of the defined keywords search strings and relevant to the field of the research. Moreover, it was noticeable the different level of relevance to the topic. To eliminate the non-relevant texts, the introduction and the conclusion of each paper have to be checked for the empirical data and any connections and links which might relate to DevOps core values based on the evaluation criteria discussed in the previous section. Table 10 displays the number of papers chosen from each database after conducting the second level selection criteria.

Table 10. Number of selected papers after initial search

Database Name	Number of selected papers
ACM	30
Wiley Online L	2
Emerald	1
IEEE	13
Proquest	2
Science Direct	4
Springer	0
Total	52

52 paper are selected, but still, the papers should be sorted to perform data extractions to get more accurate results (check figure 4). To achieve that, each paper should be evaluated based on assessment quality questions defined in the previous section to sort the papers based on the quality and the level of contribution. The assessment quality questions mentioned in table 11:

Table 11. Assessment quality questions.

Assessment Quality Question	YES	PARTILY	NO
paper aims are well described	28	22	2
RQ1 or RQ1 mentioned or discussed in the Text	15	26	11
The literature Review is clearly structured	23	22	7
DevOps is clearly the core of the discussion	38	14	0

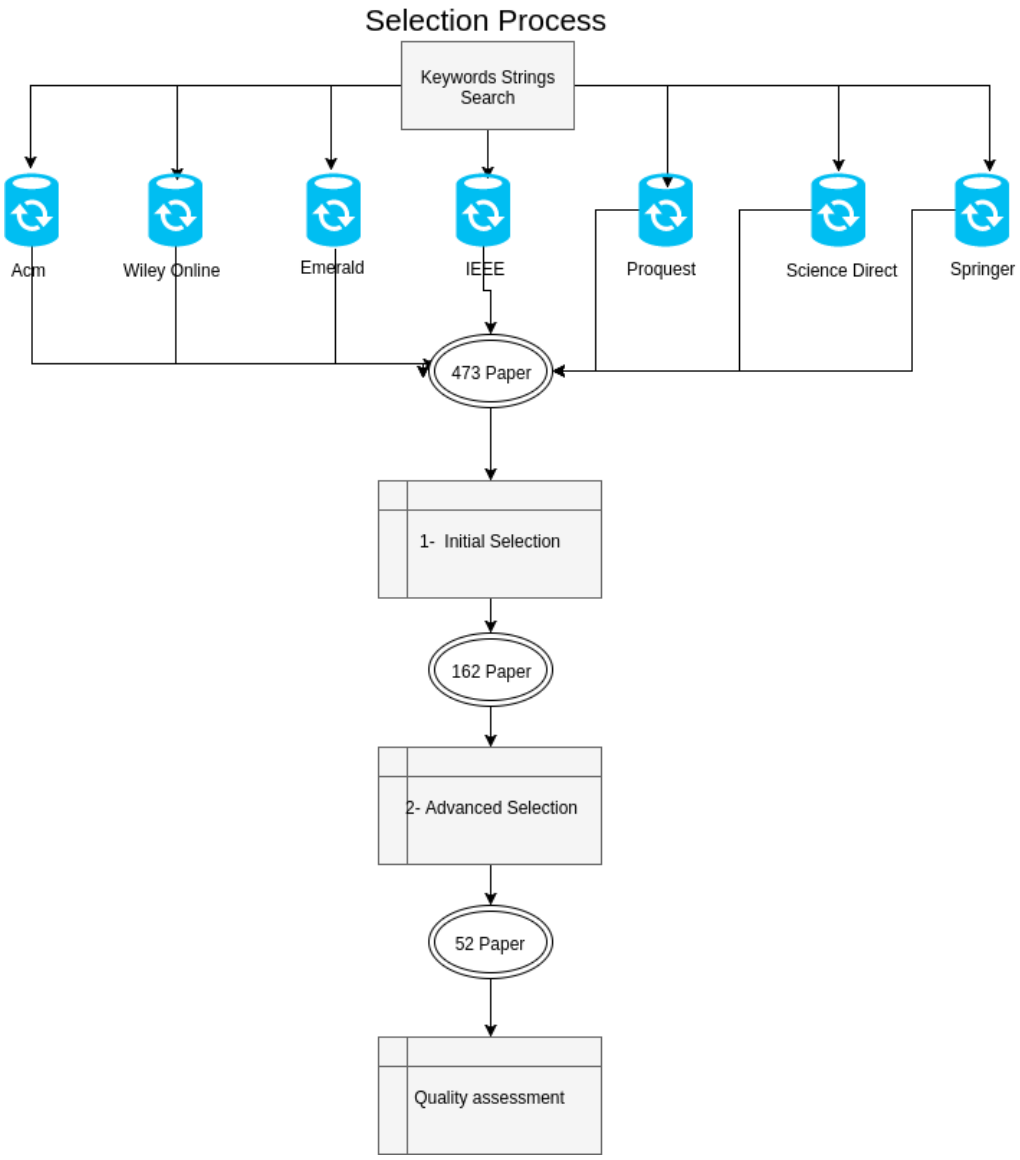

Figure 4. Data selection process.

3.5 SLR Analysis

3.5.1 Selected papers

Check appendix 1

3.5.1.1 Publication year

Since DevOps is a relatively new concept, the resources of the publication year should be limited between the years 2009-2017. The massive majority of the studies related to DevOps published after 2011. Figure 5 shows the number of selected papers per year.

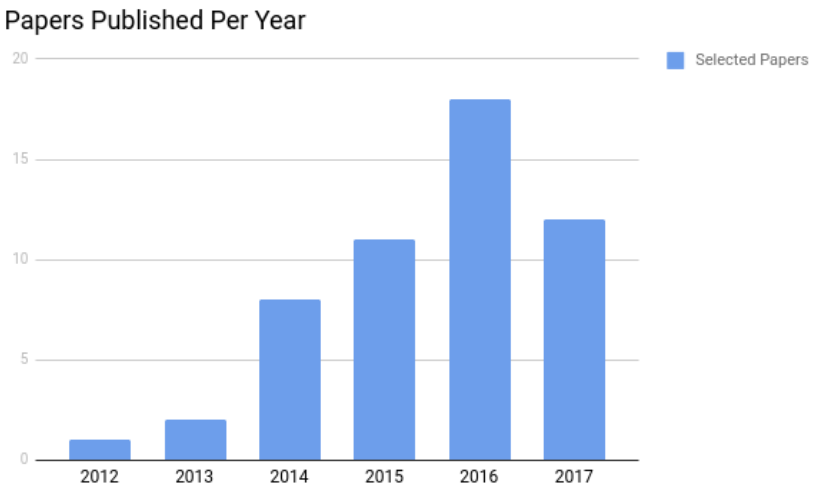

Figure 5. The number of selected papers years of publishing.

3.5.2 Papers analysis

To make the analysis of the paper easier, the papers classified based on correspondence to one of many of DevOps core values culture, automation, measurement, and sharing. Such a classification helps to draw more accurate conclusion and model which describes the reported tools and practices related to each one of DevOps core values CAMS followed in DevOps. Table 12 shows the papers and their relevance to one or more of DevOps core values classifications.

The analysis is separated into four sections, each section reflects one of DevOps core values CAMS. One additional classification is set to describe the tools used in DevOps. The first part comprises the current practices, strategies reported in the literature, the second part

describes the tools. After listing the categories, a discussion will describe the practices, strategies, and tools reported in the literature. Few papers didn't have a clear structure or clear mentioning of tools or practices, but they describe general concepts related to DevOps, the researchers didn't eliminate these papers, but they were not mentioned in the table of SLR relevant papers to one of the DevOps CAMS.

The data is extracted and classified based on the significance to one of DevOps core values, table 12 demonstrates the papers which contain associated information which contributes to the specific classification.

Table 12. Papers related to CAMS values.

DevOps CAMS	Related references
Culture	[SLR 2] [SLR 3][SLR 4][SLR 5][SLR 13][SLR 18][SLR 19][SLR 22][SLR 23][SLR 25][SLR 31][SLR 32][SLR 33][SLR 35][SLR 41][SLR 43][SLR 45][SLR 50][SLR 51][SLR 53]
Automation	[SLR 1][SLR 2][SLR 3][SLR 4][SLR 10][SLR 21][SLR 27][SLR 30][SLR 36][SLR 40][SLR 41][SLR 45][SLR 48][SLR 50][SLR 53]
Measurement	[SLR 1][SLR 12][SLR 15][SLR 16][SLR 26][SLR 27][SLR 30][SLR 36][SLR 44][SLR 45][SLR 50][SLR 53]
Sharing	[SLR 2][SLR 5][SLR 7][SLR 9][SLR 17][SLR 18][SLR 21][SLR 34][SLR 35][SLR 37][SLR 45][SLR 50][SLR 53]

3.5.2.1 Culture

19 papers related to culture, one of DevOps core values and reflect the practices reported in the literature (check appendix 2). Cultural values in DevOps context are related to the individual's attitudes and the way they communicate with their colleagues within their team and other teams. DevOps culture related values focus more on collective performance evaluation to assess the overall performance of the teams. Additionally, DevOps aims to establish a high level of trust among teams and individuals by encouraging effective communication and mutual learning, since it creates a solid foundation to DevOps culture. Establishing DevOps cultural foundation among teams make it easier to expand open to changes mentality, individual responsibility, and respect among team members [33].

Data extraction conducted to analyze, classify and report the practices and tools discussed in the chosen papers which correspond to the cultural aspects of DevOps. **Appendix 2** classifies the cultural related practices.

3.5.2.2 Automation

15 papers of the selected papers discusses automation along with its tools and best practices. Automaton supports DevOps ecosystem by allowing organizations to automate their testing, deployment, integration...etc. Additionally, Automation helps the organization to reduce time and resources consumption on repetitive tasks and increases the efficiency of handling different tasks by focusing on consistency. Automation gained popularity when the software quality and speed to market became an essential requirement for the success in the industry, many tasks were repetitive and consume a lot of time, effort, and resources. Thus the need for automation became necessary. DevOps support and enhance continuously building, deploying, testing configuring..etc. From there the automation became one of DevOps values due to the support it gives the continuous ecosystem DevOps aim to achieve. Table 13 list the automation practices mentioned in the literature [6][82][83][84].

Table 13. Automation practices reported in the literature.

No	Practices Related values	Additional details	paper/s
1.	Continuous software quality assurance monitoring	Code smell removal and detection	[SLR 1]
2.	isolating running services	host environment through either virtualization or operating system specific methods of containerization. I	[SLR 2]
3	Continuous analytics	For business needs	[SLR 3]
4	Automate the collection of the operational data	To monitor the application state in production	[SLR 4]
5	Release manager	Building the a deployment pipeline using the right architecture will improve release time	[SLR 21]
6	Repository Archival		[SLR 36]
7	Storage Cleanup		[SLR 36]
8	Management Systems Automation		[SLR 36]
9	Temporary Data Cleanup		[SLR 36]
10	Exception or Violation Analysis		[SLR 36]
11	Automated Verification	automated compliance test suite using the Fabric Python SSH (Secure Shell) library	[SLR 40]
12	Automated Bug Tracking		[SLR 48] [SLR 36]

3.5.2.3 Measurement

Measurement plays a significant role it plays to help improve the code's quality and performance. Measurement helps address the level of quality. To improve, it should be able to measure first. Software measurement helps to identify vulnerability detection, quality assessments, productivity enhancements compliance management and development practice improvements. Table 14 list practices reported in the literature to enhance measurement, one of DevOps core values.

Table 14. Measurement related practices reported in the literature.

No	Practices Related values	Additional details	paper/ s
1.	quality assurance monitoring	To help developers to detect code smell	[SLR 1]
2.	Measurement should focus on essential problems	Measurement should have an aim behind it. Any meaningless measurement should be avoided	[SLR 1]
3	Enhance Built-in quality mentality.	Tools to closely monitor quality during production.	[SLR 12]
4	Implementation of feedback mapping	the static view of code artifacts, depicted as a graph structure, is combined with the dynamic view of runtime information.	[SLR 15]
5	Implement Realtime User Monitoring (RUM)	It helps to measure and monitor metrics and provide real time data	[SLR 16]
6	Performance evaluation tool (PET)	managing measurement data independent of the data collection software	[SLR 26]
7	Analyzing audit logs	identify wrongful data usages	[SLR 36]
8	Analyzing execution logs	identify potential problems such as intrusion attacks, illegal accesses, infrastructure hitches, and capacity constraints	[SLR 36]
9	Setup Measurement friendly testing environment		[SLR 44]

3.5.2.4 Sharing

Sharing, DevOps ecosystem depend on the attitude of the individuals and their relationships with their colleagues or members of other teams. Sharing as DevOps values focus on two main factors, sharing personal learning and sharing project information. All the collected data need to be disseminated to learn to address the failure to avoid it in future projects. Sharing is not just telling facts, it is transferring ideas across teams. Table 15 list the practices related to sharing.

Table 15. Sharing related practices reported in the literature.

No	Practices Related values	Additional details	paper/s
1.	Collaborative services and responsible teams	Each services considered a standalone process, and all the services communicate with HTTP/REST API	[SLR 2]
2.	DevOps Team	This team acts as an integrator between these teams to consolidate work together and knowledge sharing.	[SLR 5]
3	Ops team monitoring dev team	Coaching and helping developers to write operational aspects of code, for example writing provisioning code	[SLR 5]
4	Logs	Logs can bridge the gap and share critical information between dev and ops teams. log statements and other runtime information, data visualizations	[SLR 7] [SLR 9]
5	TOSCA and IasC	Infrastructures that support sharing	[SLR 17]
6	Sharing status	Sharing the status of projects, services between teams will reduce configuration time	[SLR 18]
7	Frequency of communication	(decide later)	[SLR 34]
8	The direction of communication	Vertical vs horizontal	[SLR 34]
9	Communication modality	the method used to transmit information.	[SLR 34]
10	Documentation	Different Types	[SLR 34]
11	Integrated documentation builds		[SLR 37]

3.5.2.5 Tools

50 tools are identified mentioned in the literature (check appendix 3). DevOps tools help stakeholders to monitor, implement, deploy, automate and analyze each process through the journey of DevOps. Tools assess all DevOps teams, for example, the ops team to examine the health of the infrastructure and investigate any application changes. Tools assess the process of measuring the impacts by generating and evaluating a detailed metrics which could be shared with specific stakeholders. The usage of the tools can be classified into 8 different categories [46]:

- Continuous integration
- Continuous management
- Deployment
- Build
- Monitoring
- Testing
- Code analyzer
- Repository

3.5.2.5.1 Repository Management Tools

A repository manager is an application designed to manage repositories of binary components. It helps the development teams by managing different states of code, add features, delete features, versioning, and manage access to the code repository by setting up a verification process. Some repository management tools help in deployment processes. There are different repository management tools, each one serves a different purpose, there is open source one's like GitLab and some are paid [78] [71].

3.5.2.5.2 Continuous integration Tools

Continuous Integration tools help the developer teams to integrate code into a shared repository among different teams several times a day or week. The tools also allow a verification process by an automated build to allow teams to catch/detect bugs/problems as early as possible [41].

3.5.2.5.3 Configuration Management Tools

Configuration management Tools helps dev and ops team to maintain and establish a certain consistency of a particular product's performance, attributes and its functional state with its requirements, design, and operational information throughout its life [85].

3.5.2.5.4 Deployment Tools

Deployment tools help to manage code change and upload to the hosting servers, it manages the process through the entire pipeline and put it into production automatically, resulting in many production deployments every day [41].

3.5.2.5.5 Build Tools

Build tools used to create executable apps out of the source code (Example: Jar in Java). Build tools link, package, compile the code and transform it into executable form. In DevOps, build tools used to manage different dependencies, compile code, package the code and production deployment [41].

3.5.2.5.6 Monitoring Tools

Monitoring tools are essential for DevOps, so the information monitored help when a crucial decisions needed to ensure the performance and the efficiency of the services. Each tool has a specific area of focus such as infrastructure, performance, logs...etc. [41].

3.5.2.5.7 Code Analyzing Tools and testing

Code analyzes tools automate the process of finding defects and security flows within the source code, also, code analyzing tools can ensure the code quality and discover any violation of defined standards[11][46].

4 INTERVIEW

The interviews give a practical mapping between the tools and practices in DevOps and it helped to address the gaps from SLR. All the interviewees are DevOps professionals who work and manage DevOps teams in the industry. The interviewees preferred to use email interview style, because it was more convenient for them, and after that, wrapping and summarizing the interview topic over the phone or Skype. All three interviewees received the questions plus a brief summary of the thesis topic by email, some interviewees asked for clarifications about certain terms and details about the questions. It took 4-5 weeks to finish the interviews.

The interviews added insights and showed the practical processes on how practices and the tools of CAMS DevOps values are supporting DevOps ecosystem from a practical point of view. Additionally, on the values individual level, the interviews clarified few practices mentioned in the literature and added new practices. Regarding DevOps´s tools, most of the tools used in the industry reported in the SLR.

The interviews mentioned that DevOps is being implemented if the client's request to use it explicitly due to financial reasons. Because not all clients see the need of DevOps, The process to implement DevOps divided into:

1. Client Service: automation, tools, processes, scripting

2. Internal Organization: defining best standards, tools to use, community of practice and shared experiences.

One interviewee mentioned that his organization is still adopting DevOps tools and practices, and they made huge steps within their organization to understand DevOps ecosystem along with an implicit agreement that stands on the principle of using collaboration as a key mean to achieve success. The trial-and-error technique is still followed, but the process is evolving, with a constant quest for the right tools, the right processes, and the right people.

Proposed Questions:

1. To which extent do you think DevOps is being implemented in your organization?

2. How did your organization break the interdepartmental silos? Especially between Dev and Ops teams?
3. How does your organization implement culture and sharing values of DevOps?
4. Culture and Sharing are important values of DevOps, what are the tools used in your organization in order to enhance them? (what are the tools that facilitate those values)
5. To which extent is automation is being used, and what processes have been automated?
6. What are the tools used in automation?
7. Does your organization implement measurement practices?
8. What exactly is being measured?
9. What are the impacts of those measurements?
10. What are the practices followed in your organization to implement measurements?
11. What are the tools used in measurements?

4.1 Culture and Sharing

4.1.1 Practices

Remove the silos between departments the organizations by using the right process to include/cover the needs for all teams involved, agile methodologies with software development. Another organization decided to permanently include one of the team members in all other teams and giving these teams the chance to discuss how things could improve from the development and operations standpoints. The key is to show to the rest of the teams the importance, and value-added that each participant can provide by:

- Clarify team needs
- Identifying manual processes
- Automating deployments
- Reducing code testing/validation/QA time
- Sync all development stages.
- Sprint planning/grooming of tasks and defining the right priorities and criticality.
- Using the right tools to optimize/improve the current work and tasks.

The interviewees mentioned that they share their experiences with the growing/mature community of practice for DevOps worldwide. They have recurrent community group

meetings (virtual conference call), have documentation repositories for best practices, use cases, standards, aligning discussing countries success stories and shared lessons learned from all clients. There is a practice of mentoring from seniors to lower role mentees, in order to improve career paths an expertise with all resources around the different countries/locations.

4.1.2 Tools

Many tools are used to enhance the practices of DevOps to support:

- Community Repositories for DevOps teams worldwide
- Trainings for technologies and best practices, remote or onsite in different countries
- Forecast DevOps standard for tools, recommendations and best practices (emails newsletters to the community)
- Open discussions of use cases, lessons learned and current situations for specific clients.

And the tools used to support these practices and facilitate the communication of DevOps individuals locally (within the company) and globally:

- Conversations: Slack.
- Code: Git.
- Knowledge: Confluence.
- Project Management: Jira.

4.2 Automation

4.2.1 Practices

The interviewees addressed the importance of automation within DevOps ecosystem, and there are many areas where automation can apply, the main areas are related to manual/repetitive processes such as Code Deployment. It is the most common one. But it also extends to infrastructure builds in cloud services, creation, auto-scaling, self-healing, QA testing automation, and other shared services for the teams: logs tools, status pages, self-service tools (get backups and latest app versions, automatic notification for specific actions), servers provisioning, continuous integration, continuous delivery, application deployments, collaboration tools integration: code repository-build process-chat tools

(someone needs a version branch code, trigger a build, and final stage/success is notified to a chat room or email message).

4.2.2 Tools

Tools reported by interviewees are listed on appendix 4.

4.3 Measurement

4.3.1 Practices

The interviewees mentioned developing metrics for client improvement before/after DevOps processes implementation to help to track the value added, time reduction and areas to improve such as:

- Deployment time.
- Effort used to perform a task.
- Resources involved in a specific task.
- Costs associated to the improved task/process before/after of the DevOps intervention.

These measurements helps to:

- Client satisfaction.
- DevOps visibility across the organization and the value added to the process.
- Costs reduction can be visualized in a quicker way.
- Organization/Client to be willing to continue spending more resources in DevOps work.
- Trust in the work done to continue expanding the same approach to new areas/projects.

4.3.2 Tools

Tools involved from the beginning on the different projects like Jira, Confluence, and internal tools for mapping results and shared across locations. Which can help to track status and stages for the tasks performed and improved (tickets systems, document standard processes output results, track fail and success tasks) Proper documentation of the projects, the statuses of previous-current and future expectations?

5 MODEL

The model (figure 6, 7, 8 and 9) represents a set of practices and tools to enhance DevOps core values CAMS, the model separated into four sections, each section illustrates one of DevOps value's practices and tools. All information collected using SLR and interviews with DevOps practitioners. Tools reported in the model are not fixed, each organization utilizes a different set of tools to accomplish similar purposes.

In the model, the first column lists the stated cultural related practices. Some practices reported in the literature or stated by DevOps practitioners are using different words, so it was re-paraphrased to illustrate the most suitable meaning. On the other hand, the tools used to support the cultural aspects of DevOps differ from company to another and from literature document to another. However, more or less the tools perform similar tasks to support similar practices, for example, Git used in version control, but CVS also performs similar tasks, but it is a different tool.

The second column lists the automation related practices along with the most common tools. It is noticeable to the reader that continuously is the heart of automation. All automation practices support the teams to automate as much as repetitive work as possible to reduce effort, time consumption and eliminate any errors. There are plenty of tools and software to automate different processes. It is wiser for the teams to determine what are the process needed to be automated, and upon that, the tools are chosen.

Similarly, the third column lists measurement applied practices. The core of measurement practices is tracking and analyzing. Successfully analyzing data enables the stakeholders to make better decisions based on accurate predictions. Like automation tools, measurement tools vary from company to another. What needs measuring determines the most convenient tools.

The final column list the sharing practices and tools. Sharing allows better data access to the right stakeholders, unlike automation and measurement, sharing tools are not particularly related to software tools, it could be physical communication, seminars, meetings... etc.

5.1 Model Discussion

5.1.1 Culture

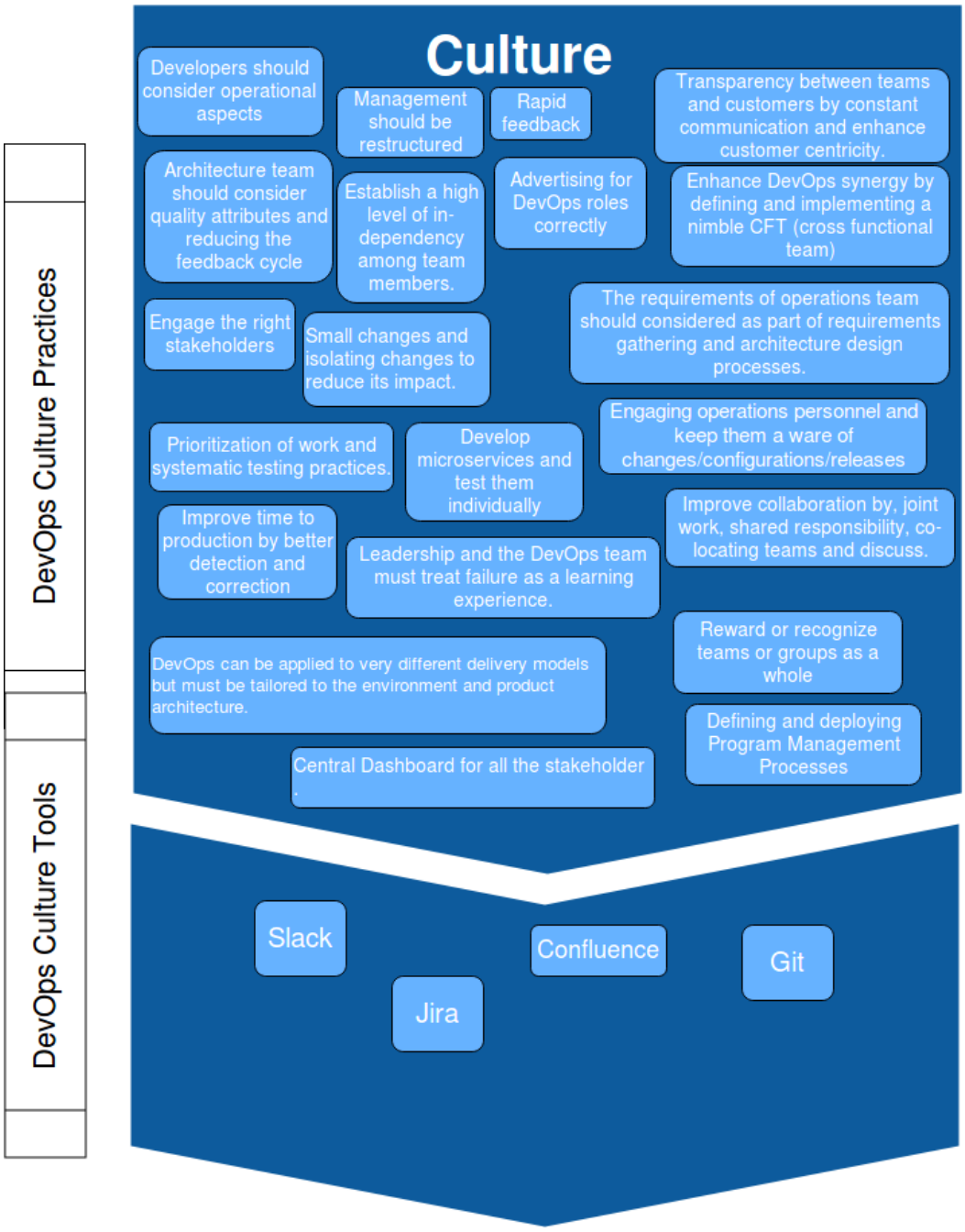

Figure 6. Culture tools and practices.

Depending on the architecture of the system, some applications should be developed in a different way to ensure independence. For example, in systems that require microservices architecture. The **microservices should be developed individually** by team and subteams. Teams should ensure that microservices are collaborating. Such approach eases the process

of deployment for the dev and ops teams and makes the development and production environment more workable, stateless and reproducible [10].

Developers should take more responsibilities, Developers should be able of developing a system while keeping in mind the operational aspects and support the architectural changes on the development and operational sides. When dev team is given more responsibilities, the code they produce becomes more operations friendly, which reduces the failure impacts of system configurations and deployments [79] [80] [81] [44].

Reducing the feedback cycle time. When architects should reduce the feedback cycle while designing the system to achieve the desired level of quality attributes. Many techniques and methodologies could mean to save, transfer the feedbacks such as aggregating logs. Reducing the feedback cycle time allows different stakeholders to be updated if any changes and failures occurred, which allows the corresponding teams to act [80] [81].

Independency among teams and team members. Establishing a high level of independence requires several steps, starting from splitting modules and components of the application into vertical level layers while architecting the system increases independency among teams. Each team responsible for implementing the assigned module or component [80]. On the other hand, **Isolating changes** can significantly reduce the impact of changes, since each change can be fixed and repaired independently without complexity [80], also making small changes enables rapid learning and innovation [57]. **Ops team requirements should be taken into consideration** while architecting the process, doing so allow the ops teams to manage, configure, analyze and act when failures happen, so **ops team should have awareness** of changes [80] [43]. Additionally, **Co-locating teams,** allow teams and individuals alike to collaborate and discuss, which improves the collaboration and communication within an organization [81].

When an organization makes the transformations of their methodology to adopt DevOps, **restructuring the management** is important along with gaining the skills needed to enhance their transformation policies [47]. Furthermore, **transparency between teams and customers** should be enhanced to understand their needs and study their reactions, this helps teams discover any bugs and defects in the system. One technique could be used is by

accessing customer's environment and data to create better transparency along with real-time estimations, in addition, customer focus should be brought to the application and customer management to reinforce customer centricity [50] [58]. **Prioritization of work** and systematic testing practices improve the workflow among different teams by clarifying priorities, which leads to better management, engagement, and communication between teams [50] [58].

Engaging the right stakeholders and making a closer collaboration with them to announce intents and agree on risk mitigation policies reduce any misunderstandings which might arise in any stage of DevOps and meet business requirements associated with balancing velocity and risk. [58].

One of the main factors that make organizations consider adopting DevOps is that it improves time to market. Enhancing **better detection and correction** policies improve time to market [58]. However, **rewarding and recognizing teams or groups** and avoiding individuality promote the individuals and teams to free their mind from "me" mindset and improve teams and individuals collaboration [58]. The leadership and the DevOps team must **treat failure as a learning experience**, and any failure could be avoided in future projects [58].

DevOps must be tailored to the environment and product architecture to create sufficient and high-quality models by choosing the right set of tools and hardware to develop the system [66]. Boosting the communication and collaboration between dev and ops team can be accomplished by **implementing program management system**. Implementing central **dashboard for the stakeholder** allow real-time performance monitoring which leads to better decision-making [68]. Releases to production is a critical process to ops team, **creating a cross-functional team** from dev and ops teams increase the sufficiency and reduce resources consumption [68].

5.1.2 Automation

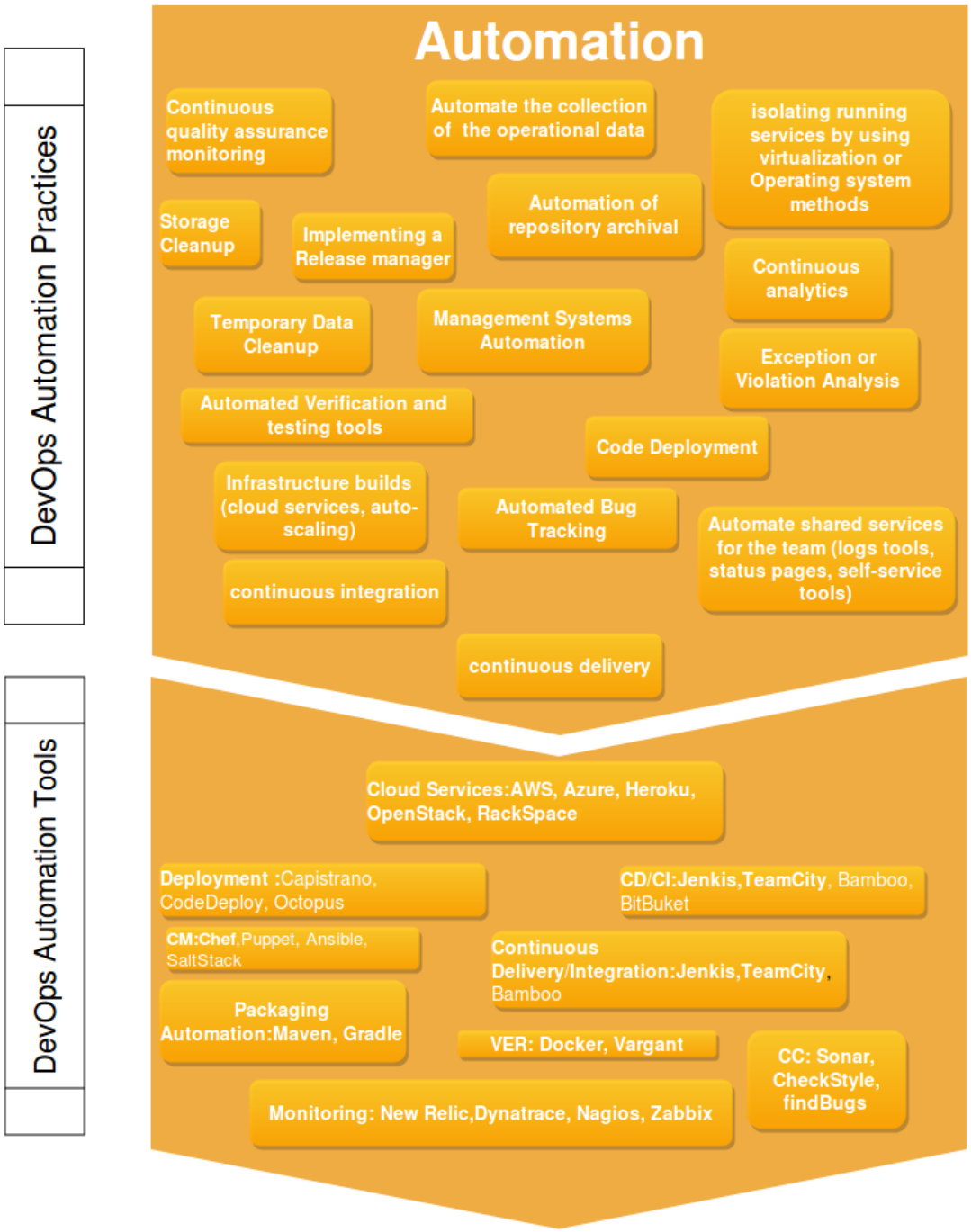

Figure 7. Automation tools and practices.

Continuous software quality assurance monitoring is an example of automation implementation, during SQA monitoring automation process the ops and dev teams can detect and bugs smell in the code and act. The system keeps monitoring the status of the application until it detects a fault and report it in real time and add the detected issue to issue tracker [78]. **Services Automatic Isolation,** is another example of automation combined with virtualization, It is important for the service to be statelessness on the servers and

clients, this can achieve isolating the services by using virtualization techniques in order to virtualize a specific operating system using containerization [10]. The use of automation is not limited to dev or ops teams, **continuous analytics** can benefit business department or help a company to achieve their business aims and objectives [79]. Taking into consideration the business needs of organizations by implementing continuous analytics in the IT infrastructure, so the business team could monitor, collect and get info in real time to meet the business aims.

Ops team should always know any changes which might occur to the system due to the new releases, code changes...etc. So, the need for a way to **automate the collection of the operational data** to support mentoring and measurement along with issues detection, so ops team can act as soon as possible to reduce the impact (defects ex. over traffic) [80]. In DevOps ecosystem, continuous deployment and releases might cause a conflict between production releases if it was not managed properly, an **automated release management** can be used to sort the release in the pipeline and help to reduce the release time [46].

The code repositories, should be managed, contained and sorted, sometimes it may cause issues in term management and performance. **Repository Archival** process checks the current status of the content, size, and performance of the repositories and makes sure the older versions to be archived [61]. Similarly, **data storage cleanup** and **temporary data cleanup** can be done through automation, the application uses repositories to execute tests tasks, to complete the tasks it may take hours and the system should avoid filling up storage to keep the performance and reduce energy consumption [61]. Companies may use different management systems, and it became difficult to manage all the IT assets spread across many management systems, management systems automation helps to tackle management systems related issues [61].

Exception or Violation Analysis can be automated to detect any potential problems such as intrusion attacks, illegal accesses, infrastructure hitches, and capacity constraints, the automation process could address these issues by analyzing the retrieval audit logs and execution logs [61]. However, another way that could impact the security is **automated verification**, this automation used to check if the client has a similar ssh key to verify the

access (GitHub as an example) [65]. **Bug tracking** automation helps to update the status of the bugs detected within the system [73].

5.1.3 Measurement

Quality assurance monitoring is an example of a measurement to check if the code is following the defined standard, QS monitoring can be automated to reduce resources consumption and faster defects detection [78]. The teams should know what they supposed to measure to avoid extra consumption, the **measurement should focus on essential problems and keep away from measuring non-related artifacts** [78]. Furthermore, different DevOps teams should focus on enhancing Built-in quality mentality, to measure and monitor the quality of the development and production environments [37].

Implementing feedback mapping by linking a set of operational data marks to nodes of the current artifacts that exist in the code generates a code view which gives the developers a way to inspect the code artifacts with data from real production traces [40]. **Real-time User Monitoring** (RUM) technique makes it possible to measure and monitor metrics and provide real time data for dev and ops teams [41].

Similarly, **Performance evaluation techniques** (PET) make it easier in managing measurement data independent of the data collection software. Since monitoring software create a high volume of data. PET provides analytics to check and refine measurements from many tests [51]. On the other hand, **Analyzing audit logs and execution logs** help to address if the data are being used wrongfully and identify any issues like attacks, illegal accesses, infrastructure hitches, and capacity constraints [61]. Moreover, **setting up a measurement friendly testing environment** (machinery, the services, virtualization and test lab management tools) that contains a fast feedback delivery, allows the corresponding teams to act accordingly and measure the quality [69].

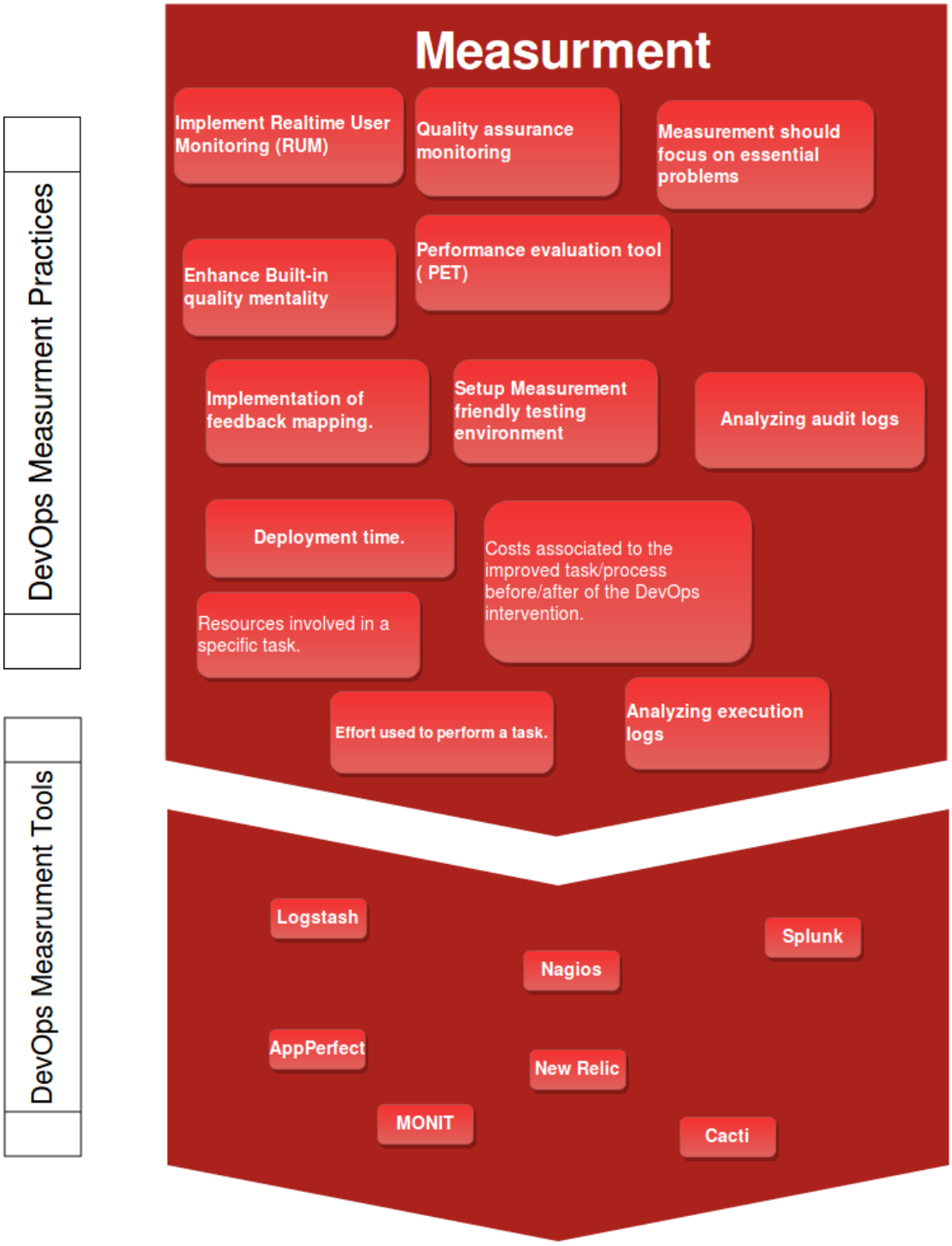

Figure 8. Measurement tools and practices

5.1.4 Sharing

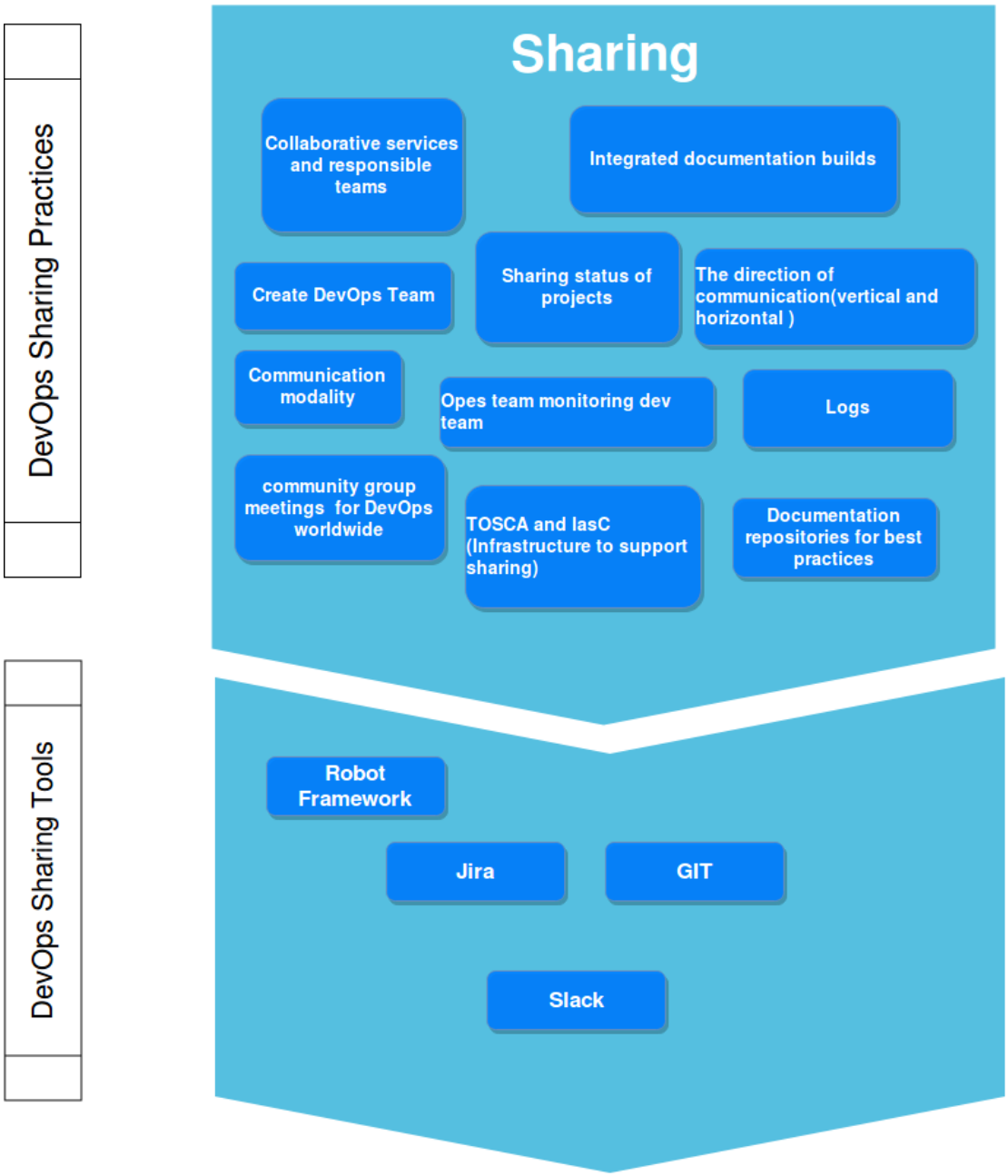

Figure 9. Sharing tools and practices.

Sharing in DevOps included sharing facts, data, and experiences. **Collaborative services and their responsible teams** is an example of how to share data between services through pre-defined rules and architectural practices. HTTP/REST API is a way to communicate between services, where teams follow REST API architecture and define endpoints to use it in other teams [10]. However, **creating a DevOps team** within an organization might

increase the amount of collaboration between different teams, this team acts as an integrator between these teams to merge work and knowledge sharing [81]. However, some organizations follow other practices, where Opes team monitor and coach dev team by helping them implement operational aspects of the code, for example writing provisioning code [81].

A common way to share technical information within an organization is Logs, Logs can bridge the gap and share critical information between dev and ops teams, the logs can include a numerous amount of information about different operations such as log statements and other runtime information, data visualizations [32][34]. Besides, some organizations prefer to implement an infrastructure that enhances sharing such as TOSCA and IasC [42].

Sharing the projects, services status among the involved stakeholder reduce configuration time, thus reduce resources consumption and enhances sharing [43]. Moreover, DevOps teams should consider the type of communication and sharing methods followed in the organizations, and analyze the variable that might affect the current way of sharing. **The frequency of communication,** Direction of communication and **modality** of communication is all variable that should be taken into consideration while developing a new DevOps sharing strategies [59]. Documentation is still one way of sharing, there are different ways of documenting in the DevOps and in the industry, some still uses Text format files, JSON, XML.. etc [59]. However, integrated documentation is another way to enhance sharing by **integrating the documentation** with the code and other operations aspects [62]

5.2 Model Validation

One of the last steps of DSR is to validate the artifact created, the validation will help to refine the artifact if needed and check whether it is fulfilling the purpose it was to be designed for. Experts opinions is the method used to determine the validity of the model, the insights and the suggestions from the experts will help to:

- Validate the accuracy of the model.
- Check the quality of the model.
- Validate the structure of the model.
- Check the understandability of the model.
- Check for flaws and errors in the model.

- Check if the model lack of important information.
- Overall opinion about the model and its benefit.
- Provide any additional information if necessary.

5.2.1 Experts Opinions

All the experts (check table 16) are currently working in the industry.

Table 16. Names and Roles of the Experts

Name	Role	Company	Country
Mr. Timo Stordell	DevOps Consultant	Eficode oy	Finland
Mr. Mauricio Araya	Architect/DevOps Engineer	Qwinix Technologies	Costa Rica
Mr. Orlando Escalante	IT SysAdmin Engineer/ DevOps Specialist	Q6 Cyber	Costa Rica

The experts have received the first version of the model; the model created after conducting SLR and interviews to extract information related to DevOps core values CAMS. The model summarizes the practices and tools used in each value of DevOps. The experts' opinions shared many similarities, however, each expert provided slightly different insights on different aspects of the model. The suggestions reflected on the model to optimize it and refine it to suit the given insights.

Mr. Timo and Mr. Araya have agreed on the model in general and its validity. However, they have highlighted several points to consider:

- Automated verification should highlight the need for testing tools.
- Robot framework should be added to the tools in the model from automated verification and sharing perspective.
- Mr. Timo suggested a model created by his team from processes and technical perspective (check figure 10) to use it as a reference to the model.
- The model require some clean up, design wise to look more consistent.
- Some parts of the model are hard to read.
- Mr. Araya suggested to split the model into several figures to make it easier read.

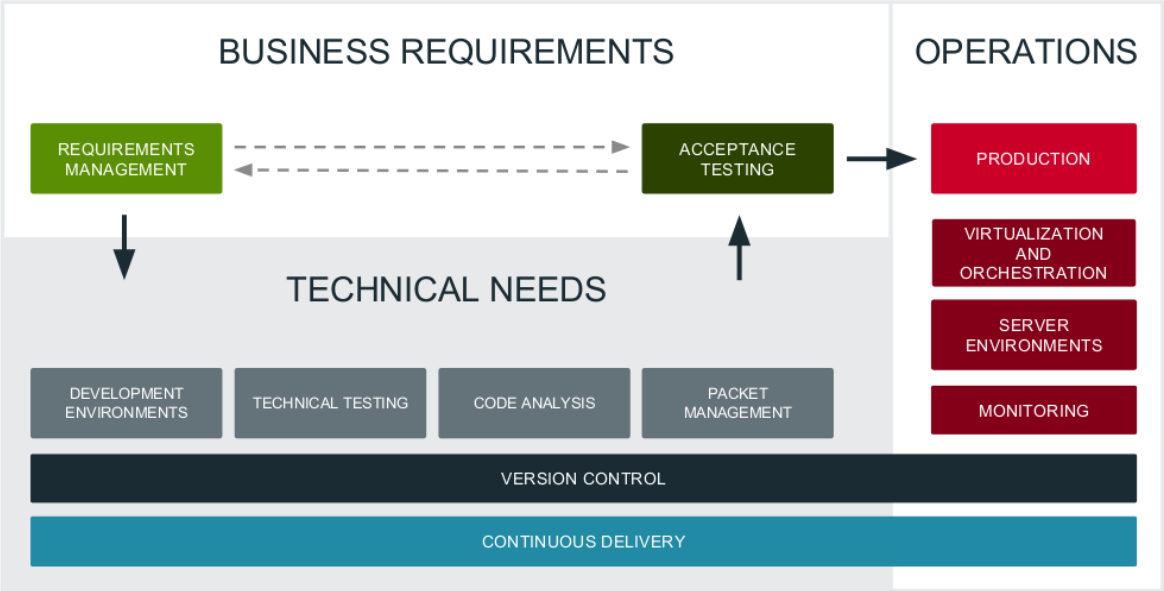

Figure 10. DevOps Culture to Organization. [96]

The suggestions taken into consideration and used to reform the model (check appendix 6). The model refined and split into 4 main figures, each figure represents one DevOps core values.

6 CONCULSIONS

The model in its current state lists the practices and tools regarding each core value of DevOps. It gives technical practices which reflect on DevOps ecosystem to enhance its ability and adaptability to improve software development processes. In the 3rd chapter, the practices discussed comprehensively, and then the tools that facilitate the implementation of the practices listed (check appendix 3).

In general, the literature about DevOps is particularly lacking any clear standard to define the practices and the tools, even the definition of DevOps vary from one journal to another. However, the literature gave a comprehensive idea about concepts related to DevOps. Some articles describe DevOps related concepts but with no direct mention that it is linked to DevOps. This issue occurs due to the fact that DevOps as a term is relatively new that resulted in a confusion at the beginning of conducting SLR. Data extracted in SLR listed in a way to makes it easier to understand, by demonstrating the different core values of DevOps, and how these values can be enhanced by following a set of practices.

The interviews are the second method of choice to collect the data, DevOps practitioners gave evaluable thoughts about the current state of DevOps in the industry. The practices and tools reported in the literature and interviews matches, but interviewees added more practical and technical practices. Automation was the main area of confusion, due to the uncertainty about its practices and tools. Automation practices and tools differ from company to another, and it is affected by the domain of the industry.

Based on the practices and tools listed and collected from the literature, a model created to represent and summarize the findings. However, as a part of DSR, the model needs validation for refinement and improvement. Experts have validated the model, but they added several insights regarding tools, practices and the model structure. The model updated and restructured to fit the suggestions.

REFERENCES

1. Cois, C. A., Yankel, J., & Connell, A. (2014, October). Modern DevOps: Optimizing software development through effective system interactions. In Professional Communication Conference (IPCC), 2014 IEEE International (pp. 1-7). IEEE.
2. Hamunen, J. (2016). Challenges in adopting a Devops approach to software development and operations.
3. Erich, F. M. A., Amrit, C., & Daneva, M. (2017). A Qualitative Study of DevOps Usage in Practice.
4. Chef (2017). Definition of DevOps. Accessed in : https://pages.chef.io/rs/255-VFB-268/images/EmbraceDevops.pdf
5. Hussaini, S. W. (2014, October). Strengthening harmonization of development (dev) and operations (ops) silos in it environment through systems approach. In Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on (pp. 178-183). IEEE.
6. Cruzes, D., Diel, E., & Marczak, S. (2016). Communication Challenges and Strategies in Distributed DevOps. 2016 IEEE 11th International Conference on Global Software Engineering (ICGSE), 24-28.
7. Erich, F., Amrit, C., & Daneva, M. (2014). Report: Devops literature review. University of Twente, Tech. Rep.
8. Yiran, Z., & Yilei, L. (2017). The Challenges and Mitigation Strategies of Using DevOps during Software Development.
9. Belagatti, P. (2017). Why Organizations Fail to Adopt CI and CD - DZone DevOps. dzone.com. Retrieved 20 January 2017, from https://dzone.com/articles/why-organizations-fail-to-adopt-cicd
10. Stillwell, M., & Coutinho, J. G. (2015, September). A DevOps approach to integration of software components in an EU research project. In Proceedings of the 1st International Workshop on Quality-Aware DevOps (pp. 1-6). ACM.
11. Austel, P., Chen, H., Mikalsen, T., Rouvellou, I., Sharma, U., Silva-Lepe, I., & Subramanian, R. (2015, June). Continuous delivery of composite solutions: A case for collaborative software defined paas environments. In Proceedings of the 2nd International Workshop on Software-Defined Ecosystems (pp. 3-6). ACM.
12. Von Alan, R. H., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. MIS quarterly, 28(1), 75-105.

13. Gregor, S., & Hevner, A. R. (2013). Positioning and presenting design science research for maximum impact. MIS quarterly, 37(2).
14. Peffers, K., Tuunanen, T., Gengler, C. E., Rossi, M., Hui, W., Virtanen, V., & Bragge, J. (2006, February). The design science research process: a model for producing and presenting information systems research. In Proceedings of the first international conference on design science research in information systems and technology (DESRIST 2006) (pp. 83-106). Sn.
15. Keele, S. (2007). Guidelines for performing systematic literature reviews in software engineering. In Technical report, Ver. 2.3 EBSE Technical Report. EBSE. sn.
16. B.A. Kitchenham, S. Charters, Guidelines for Performing Systematic Literature Reviews in Software Engineering Technical Report EBSE-2007-01, 2007.
17. Paul, F. (2014). The Incredible True Story of How DevOps Got Its Name. New Relic Blog. Retrieved 8 August 2017, from https://blog.newrelic.com/2014/05/16/devops-name/
18. Novak, A. (2017). Most Companies Deploying Code Weekly, Daily, or Hourly. New Relic Blog. Retrieved 4 February 2016, from https://blog.newrelic.com/2016/02/04/data-culture-survey-results-faster-deployment/
19. [UDACITY]. (2016, Jun 6).Software Development Lifecycle [Video File]. Retrieved from https://youtu.be/8i_8HnnO83Q
20. [UDACITY]. (2016, Jun 6).Definition Of DevOps [Video File]. Retrieved from https://youtu.be/p41OYYbatgQ
21. UDACITY]. (2016, Jun 6).M is for Measurement [Video File]. Retrieved from https://youtu.be/4IXgVwxhnvU
22. B. S. Farroha and D. L. Farroha, "A Framework for Managing Mission Needs, Compliance, and Trust in the DevOps Environment," in 2014 IEEE Military Communications Conference, 2014, pp. 288–293.
23. J. Wettinger, U. Breitenbücher, and F. Leymann, "DevOpSlang–bridging the gap between development and operations," in European Conference on Service-Oriented and Cloud Computing,2014, pp. 108–122.
24. L. Evenstad, "Delivering Success with Devops," Computer Weekly, pp. 23–26, Dec. 2015.
25. Willis, J. (2012). DevOps Culture (Part 1) - IT Revolution. IT Revolution. Retrieved 14 August 2017, from https://itrevolution.com/devops-culture-part-1/
26. Cuppett, M. S. (2016). DevOps for DBAs. In DevOps, DBAs, and DBaaS (pp. 1-13). Apress.

27. [UDACITY]. (2016, Jun 6).C is for Culture [Video File]. Retrieved from https://youtu.be/dNfrYSJOfOk
28. [UDACITY]. (2016, Jun 6).A is for Automation [Video File]. Retrieved from https://youtu.be/n7K5SpMrQt0
29. [UDACITY]. (2016, Jun 6).S Stands for Sharing [Video File]. Retrieved from https://youtu.be/bC4Gyyzu7K8
30. Okoli, C., & Schabram, K. (2010). A guide to conducting a systematic literature review of information systems research. Sprouts Work. Pap. Inf. Syst, 10(26). Chicago
31. Shahin, M. (2015, September). Architecting for devops and continuous deployment. In Proceedings of the ASWEC 2015 24th Australasian Software Engineering Conference (pp. 147-148). ACM.
32. Shang, W. (2012, June). Bridging the divide between software developers and operators using logs. In Software Engineering (ICSE), 2012 34th International Conference on (pp. 1583-1586). IEEE.
33. de Fran?a, B. B. N., Jeronimo Junior, H., & Travassos, G. H. (2016, September). Characterizing DevOps by Hearing Multiple Voices. In Proceedings of the 30th Brazilian Symposium on Software Engineering (pp. 53-62). ACM.
34. Cito, J., Oliveira, F., Leitner, P., Nagpurkar, P., & Gall, H. C. (2017, May). Context-based analytics: establishing explicit links between runtime traces and source code. In Proceedings of the 39th International Conference on Software Engineering: Software Engineering in Practice Track (pp. 193-202). IEEE Press.
35. Ferry, N., Chauvel, F., Song, H., & Solberg, A. (2015, September). Continous deployment of multi-cloud systems. In Proceedings of the 1st International Workshop on Quality-Aware DevOps (pp. 27-28). ACM.
36. Austel, P., Chen, H., Mikalsen, T., Rouvellou, I., Sharma, U., Silva-Lepe, I., & Subramanian, R. (2015, June). Continuous delivery of composite solutions: A case for collaborative software defined paas environments. In Proceedings of the 2nd International Workshop on Software-Defined Ecosystems (pp. 3-6). ACM. 11
37. Fitzgerald, B., & Stol, K. J. (2014, June). Continuous software engineering and beyond: trends and challenges. In Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering (pp. 1-9). ACM.
38. Erich, F., Amrit, C., & Daneva, M. (2014, September). Cooperation between information system development and operations: a literature review. In Proceedings of the 8th

ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (p. 69). ACM.

39. Erich, F., Amrit, C., & Daneva, M. (2014, September). Cooperation between information system development and operations: a literature review. In Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (p. 69). ACM.
40. Cito, J. (2016, August). Developer targeted analytics: supporting software development decisions with runtime information. In Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering (pp. 892-895). ACM.
41. Cukier, D. (2013, October). DevOps patterns to scale web applications using cloud services. In Proceedings of the 2013 companion publication for conference on Systems, programming, & applications: software for humanity (pp. 143-152). ACM.
42. Artac, M., Borovsak, T., Di Nitto, E., Guerriero, M., & Tamburri, D. A. (2017, May). DevOps: introducing infrastructure-as-code. In Proceedings of the 39th International Conference on Software Engineering Companion (pp. 497-498). IEEE Press.
43. Bass, L., Jeffery, R., Wada, H., Weber, I., & Zhu, L. (2013, May). Eliciting operations requirements for applications. In Proceedings of the 1st International Workshop on Release Engineering (pp. 5-8). IEEE Press.
44. K?rp?noja, P., Virtanen, A., Lehtonen, T., & Mikkonen, T. (2016, May). Exploring Peopleware in Continuous Delivery. In Proceedings of the Scientific Workshop Proceedings of XP2016 (p. 13). ACM.
45. Chung, S., & Bang, S. (2016). Identifying knowledge, skills, and abilities (KSA) for devops-aware server side web application with the grounded theory. Journal of Computing Sciences in Colleges, 32(1), 110-116.
46. Krusche, S., & Alperowitz, L. (2014, May). Introduction of continuous delivery in multi-customer project courses. In Companion Proceedings of the 36th International Conference on Software Engineering (pp. 335-343). ACM.
47. Jones, S., Noppen, J., & Lettice, F. (2016, July). Management challenges for DevOps adoption within UK SMEs. In Proceedings of the 2nd International Workshop on Quality-Aware DevOps (pp. 7-11). ACM.
48. Arta?, M., Borov?ak, T., Di Nitto, E., Guerriero, M., & Tamburri, D. A. (2016, July). Model-driven continuous deployment for quality devops. In Proceedings of the 2Nd International Workshop on Quality-Aware DevOps (pp. 40-41). ACM.

49. Ustinova, T., & Jamshidi, P. (2015, September). Modelling multi-tier enterprise applications behaviour with design of experiments technique. In Proceedings of the 1st International Workshop on Quality-Aware DevOps (pp. 13-18). ACM.
50. Virtanen, A., Kuusinen, K., Lepp?nen, M., Luoto, A., Kilamo, T., & Mikkonen, T. (2017, April). On continuous deployment maturity in customer projects. In Proceedings of the Symposium on Applied Computing (pp. 1205-1212). ACM.
51. Kro?, J., Willnecker, F., Zwickl, T., & Krcmar, H. (2016, July). PET: continuous performance evaluation tool. In Proceedings of the 2nd International Workshop on Quality-Aware DevOps (pp. 42-43). ACM.
52. Guerriero, M., Ciavotta, M., Gibilisco, G. P., & Ardagna, D. (2015, September). Space4cloud: a devops environment for multi-cloud applications. In Proceedings of the 1st International Workshop on Quality-Aware DevOps (pp. 29-30). ACM.
53. Wettinger, J., Breitenb?cher, U., & Leymann, F. (2014, December). Standards-based DevOps automation and integration using TOSCA. In Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing (pp. 59-68). IEEE Computer Society.
54. Clear, T. (2017). THINKING ISSUES Meeting employers expectations of devops roles: can dispositions be taught?. ACM Inroads, 8(2), 19-21.
55. Perez, J. F., Wang, W., & Casale, G. (2015, January). Towards a devops approach for software quality engineering. In Proceedings of the 2015 Workshop on Challenges in Performance Methods for Software Development (pp. 5-10). ACM.
56. Kerzazi, N., & Adams, B. (2016, May). Who needs release and devops engineers, and why?. In Continuous Software Evolution and Delivery (CSED), IEEE/ACM International Workshop on (pp. 77-83). IEEE.
57. Denning, S. (2015). New lessons for leaders about continuous innovation. Strategy & Leadership, 43(1), 11-15.
58. Farroha, B. S., & Farroha, D. L. (2014, October). A framework for managing mission needs, compliance, and trust in the DevOps environment. In Military Communications Conference (MILCOM), 2014 IEEE (pp. 288-293). IEEE.
59. Diel, E., Marczak, S., & Cruzes, D. S. (2016, August). Communication Challenges and Strategies in Distributed DevOps. In Global Software Engineering (ICGSE), 2016 IEEE 11th International Conference on (pp. 24-28). IEEE.

60. Vassallo, C., Zampetti, F., Romano, D., Beller, M., Panichella, A., Di Penta, M., & Zaidman, A. (2016, October). Continuous delivery practices in a large financial organization. In Software Maintenance and Evolution (ICSME), 2016 IEEE International Conference on (pp. 519-528). IEEE.
61. Pang, C., & Hindle, A. (2016, October). Continuous Maintenance. In Software Maintenance and Evolution (ICSME), 2016 IEEE International Conference on (pp. 458-462). IEEE.
62. Waits, T., & Yankel, J. (2014, October). Continuous system and user documentation integration. In Professional Communication Conference (IPCC), 2014 IEEE International (pp. 1-5). IEEE.
63. Zhu, L., Bass, L., & Champlin-Scharff, G. (2016). Devops and its practices. IEEE Software, 33(3), 32-34.
64. Rajkumar, M., Pole, A. K., Adige, V. S., & Mahanta, P. (2016, April). DevOps culture and its impact on cloud delivery and software development. In Advances in Computing, Communication, & Automation (ICACCA)(Spring), International Conference on (pp. 1-6). IEEE.
65. Callanan, M., & Spillane, A. (2016). DevOps: making it easy to do the right thing. IEEE Software, 33(3), 53-59.
66. Ebert, C., Gallardo, G., Hernantes, J., & Serrano, N. (2016). DevOps. IEEE Software, 33(3), 94-100.
67. Furfaro, A., Gallo, T., Garro, A., Sacc?, D., & Tundis, A. (2016, September). ResDevOps: A Software Engineering Framework for Achieving Long-Lasting Complex Systems. In Requirements Engineering Conference (RE), 2016 IEEE 24th International (pp. 246-255). IEEE.
68. Hussaini, S. W. (2014, October). Strengthening harmonization of development (dev) and operations (ops) silos in it environment through systems approach. In Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on (pp. 178-183). IEEE.
69. Wahaballa, A., Wahballa, O., Abdellatief, M., Xiong, H., & Qin, Z. (2015, September). Toward unified DevOps model. In Software Engineering and Service Science (ICSESS), 2015 6th IEEE International Conference on (pp. 211-214). IEEE.

70. Virmani, M. (2015, May). Understanding DevOps & bridging the gap from continuous integration to continuous delivery. In Innovative Computing Technology (INTECH), 2015 Fifth International Conference on (pp. 78-82). IEEE.
71. Vaasanthi, R., Kumari, V. P., & Kingston, S. P. (2017). Analysis of Devops Tools using the Traditional Data Mining Techniques. International Journal of Computer Applications, 161(11).
72. Wang, X., Clay, P. F., & Forsgren, N. (2015). Encouraging knowledge contribution in IT support: social context and the differential effects of motivation type. Journal of Knowledge Management, 19(2), 315-333.
73. Bartusevics, A. (2017). Automation of Continuous Services: What Companies of Latvia Says about It?. Procedia Computer Science, 104, 81-88.
74. Hosono, S., & Shimomura, Y. (2017). Bridging On-site Practices and Design Principles for Service Development. Procedia CIRP, 60, 422-427. Chicago
75. Rodr?guez, P., Haghighatkhah, A., Lwakatare, L. E., Teppola, S., Suomalainen, T., Eskeli, J., ... & Oivo, M. (2017). Continuous deployment of software intensive products and services: A systematic mapping study. Journal of Systems and Software, 123, 263-291.
76. Fitzgerald, B., & Stol, K. J. (2017). Continuous software engineering: A roadmap and agenda. Journal of Systems and Software, 123, 176-189. Chicago
77. Erich, F. M. A., Amrit, C., & Daneva, M. (2017). A Qualitative Study of DevOps Usage in Practice.
78. Janes, A., Lenarduzzi, V., & Stan, A. C. (2017, April). A Continuous Software Quality Monitoring Approach for Small and Medium Enterprises. In Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering Companion (pp. 97-100). ACM.
79. Di Nitto, E., Jamshidi, P., Guerriero, M., Spais, I., & Tamburri, D. A. (2016, July). A software architecture framework for quality-aware devops. In Proceedings of the 2nd International Workshop on Quality-Aware DevOps (pp. 12-17). ACM.
80. Shahin, M., Babar, M. A., & Zhu, L. (2016, September). The Intersection of Continuous Deployment and Architecting Process: Practitioners' Perspectives. In Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (p. 44). ACM.

81. Shahin, M., Zahedi, M., Babar, M. A., & Zhu, L. (2017, June). Adopting Continuous Delivery and Deployment: Impacts on Team Structures, Collaboration and Responsibilities. In Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering (pp. 384-393). ACM.
82. Lwakatare, L.E., M?kinen, S., M?nnist?, T., Riungu-Kalliosaari, L., & Tiihonen, J. (2016). DevOps Adoption Benefits and Challenges in Practice: A Case Study. PROFES.
83. Breitenb?cher, U., Falkenthal, M., Leymann, F., & Wettinger, J. (2016). Collaborative gathering and continuous delivery of DevOps solutions through repositories. Computer Science - Research and Development, 1-10.
84. Abdellatief, M., Qin, Z., Wahaballa, A., Wahballa, O., & Xiong, H. (2015). Toward unified DevOps model. 2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS), 211-214.
85. Spinellis, D. (2012). Don't install software by hand. IEEE software, 29(4), 86-87.
86. Gregor, S., & Hevner, A. R. (2013). Positioning and presenting design science research for maximum impact. MIS quarterly, 37(2).
87. Kuechler, B., & Vaishnavi, V. (2004). Design Science Research in Information Systems. URI: http://www. desrist. org/design-research-in-information-systems/. Online.
88. Pahl, C., Jamshidi, P., & Weyns, D. (2017). Cloud architecture continuity: Change models and change rules for sustainable cloud software architectures. Journal of Software: Evolution and Process, 29(2).
89. Waller, J., Ehmke, N. C., & Hasselbring, W. (2015). Including performance benchmarks into continuous integration to enable DevOps. ACM SIGSOFT Software Engineering Notes, 40(2), 1-4.
90. Jabbari, R., bin Ali, N., Petersen, K., & Tanveer, B. (2016, May). What is DevOps?: A Systematic Mapping Study on Definitions and Practices. In Proceedings of the Scientific Workshop Proceedings of XP2016 (p. 12). ACM.
91. Perera, P., Bandara, M., & Perera, I. (2016, September). Evaluating the impact of DevOps practice in Sri Lankan software development organizations. In Advances in ICT for Emerging Regions (ICTer), 2016 Sixteenth International Conference on (pp. 281-287). IEEE.
92. Erich, F., Amrit, C., & Daneva, M. (2014). Cooperation between software development and operations: a literature review. ESEM.

93. Erich, F., Amrit, C., & Daneva, M. (2014, September). Cooperation between information system development and operations: a literature review. In Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (p. 69). ACM.
94. Colavita, F. (2016). DevOps Movement of Enterprise Agile Breakdown Silos, Create Collaboration, Increase Quality, and Application Speed. In Proceedings of 4th International Conference in Software Engineering for Defence Applications (pp. 203-213). Springer International Publishing.
95. Hussaini, S. W. (2015). A Systemic Approach to Re-inforce Development and Operations Functions in Delivering an Organizational Program. Procedia Computer Science, 61, 261-266.
96. Stordell, T. (2017). Boosting your SW development with Devops. Slideshare.net. Retrieved 12 December 2017, from https://www.slideshare.net/TimoStordell/boosting-your-software-development-20170324

APPENDIX

APPENDIX 1. SLR selected papers

No	Title	Author/s	Year	Database	Reference
SLR 1	A Continuous Software Quality Monitoring Approach for Small and Medium Enterprises	Janes, Andrea, Valentina Lenarduzzi	2017	ACM	[78n]
SLR 2	A DevOps approach to integration of software components in an EU research project	Stillwell, M., & Coutinho, J. G.	2015	ACM	[10n]
SLR 3	A software architecture framework for quality-aware devops	Di Nitto, E., Jamshidi, P., Guerriero, M., Spais, I., & Tamburri, D. A	2016	ACM	[79n]
SLR 4	The Intersection of Continuous Deployment and Architecting Process: Practitioners' Perspectives.	Shahin, M., Babar, M. A., & Zhu, L.	2016	ACM	[80n]
SLR 5	Adopting Continuous Delivery and Deployment: Impacts on Team Structures, Collaboration and Responsibilities	Shahin, M., Zahedi, M., Babar, M. A., & Zhu, L	2017	ACM	[81n]
SLR 6	Architecting for devops and continuous deployment.	Shahin, M.	2015	ACM	31
SLR 7	Bridging the divide between software developers and operators using logs	Shang, W.	2012	ACM	32

SLR 8	Characterizing DevOps by Hearing Multiple Voices.	de França, B. B. N., Jeronimo Junior, H., & Travassos, G. H.	2016	ACM	33
SLR 9	Context-based analytics: establishing explicit links between runtime traces and source code	Cito, J., Oliveira, F., Leitner, P., Nagpurkar, P., & Gall, H. C	2017	ACM	34
SLR 10	Continous deployment of multi-cloud systems.	Ferry, N., Chauvel, F., Song, H., & Solberg, A.	2015	ACM	35
SLR 11	Continuous delivery of composite solutions: A case for collaborative software defined paas environments	Austel, P., Chen, H., Mikalsen, T., Rouvellou, I., Sharma, U., Silva-Lepe, I.,	2015	ACM	36 [11n]
SLR 12	Continuous software engineering and beyond: trends and challenges.	Fitzgerald, B., & Stol, K. J.	2014	ACM	37
SLR 13	Cooperation between information system development and operations: a literature review.	Erich, F., Amrit, C., & Daneva, M.	2014	ACM	38
SLR 14	Cooperation between information system development and operations: a literature review.	Erich, F., Amrit, C., & Daneva, M.	2014	ACM	39
SLR 15	Developer targeted analytics: supporting software development decisions with runtime information.	Cito, J.	2016	ACM	40

SLR 16	DevOps patterns to scale web applications using cloud services	Cukier, D.	2013	ACM	41
SLR 17	DevOps: introducing infrastructure-as-code	Artač, M., Borovšak, T., Di Nitto, E., Guerriero, M., & Tamburri, D. A	2017	ACM	42
SLR 18	Eliciting operations requirements for applications	Bass, L., Jeffery, R., Wada, H., Weber, I., & Zhu, L.	2013	ACM	43
SLR 19	Exploring Peopleware in Continuous Delivery.	Kärpänoja, P., Virtanen, A., Lehtonen, T., & Mikkonen, T.	2016	ACM	44
SLR 20	Identifying knowledge, skills, and abilities (KSA) for devops-aware server side web application with the grounded theory.	Chung, S., & Bang, S.	2016	ACM	45
SLR 21	Introduction of continuous delivery in multi-customer project courses.	Krusche, S., & Alperowitz, L.	2014	ACM	46
SLR 22	Management challenges for DevOps adoption within UK SMEs.	Jones, S., Noppen, J., & Lettice, F.	2016	ACM	47
SLR 23	Model-driven continuous deployment for quality DevOps	Artač, M., Borovšak, T., Di Nitto, E., Guerriero, M., & Tamburri, D. A.	2016	ACM	48

SLR 24	Modelling multi-tier enterprise applications behaviour with design of experiments technique	Ustinova, T., & Jamshidi, P.	2015	ACM	49
SLR 25	. On continuous deployment maturity in customer projects.	Virtanen, A., Kuusinen, K., Leppänen, M., Luoto, A., Kilamo, T., & Mikkonen, T.	2017	ACM	50
SLR 26	PET: continuous performance evaluation tool.	Kroß, J., Willnecker, F., Zwickl, T., & Krcmar, H.	2016	ACM	51
SLR 27	Space4cloud: a devops environment for multi-cloud applications.	Guerriero, M., Ciavotta, M., Gibilisco, G. P., & Ardagna, D.	2015	ACM	52
SLR 28	Standards-based DevOps automation and integration using TOSCA.	Wettinger, J., Breitenbücher, U., & Leymann, F.	2014	ACM	53
SLR 29	THINKING ISSUES Meeting employers expectations of devops roles: can dispositions be taught?	Clear, T.	2017	ACM	54
SLR 30	Towards a devops approach for software quality engineering.	Perez, J. F., Wang, W., & Casale, G	2015	ACM	55
SLR 31	Who needs release and devops engineers, and why?	Kerzazi, N., & Adams, B.	2016	ACM	56
SLR 32	New lessons for leaders about continuous innovation.	Denning, S.	2015	Emerald	57
SLR 33	A framework for managing mission needs, compliance, and	Farroha, B. S., & Farroha, D. L	2014	IEEE	58

	trust in the DevOps environment.				
SLR 34	Communication Challenges and Strategies in Distributed DevOps.	Diel, E., Marczak, S., & Cruzes, D. S.	2016	IEEE	59
SLR 35	Continuous delivery practices in a large financial organization.	Vassallo, C., Zampetti, F., Romano, D., Beller, M., Panichella, A., Di Penta, M., & Zaidman, A	2016	IEEE	60
SLR 36	Continuous Maintenance.	Pang, C., & Hindle, A.	2016	IEEE	61
SLR 37	Continuous system and user documentation integration	Waits, T., & Yankel, J.	2014	IEEE	62
SLR 38	Devops and its practices.	Zhu, L., Bass, L., & Champlin-Scharff, G.	2016	IEEE	63
SLR 39	DevOps culture and its impact on cloud delivery and software development.	Rajkumar, M., Pole, A. K., Adige, V. S., & Mahanta, P.	2016	IEEE	64
SLR 40	DevOps: making it easy to do the right thing.	Callanan, M., & Spillane, A.	2016	IEEE	65
SLR 41	DevOps	Ebert, C., Gallardo, G., Hernantes, J., & Serrano, N.	2016	IEEE	66
SLR 42	ResDevOps: A Software Engineering Framework for Achieving Long-Lasting Complex Systems.	Furfaro, A., Gallo, T., Garro, A., Saccà, D., & Tundis, A.	2016	IEEE	67

SLR 43	Strengthening harmonization of Development (Dev) and Operations (Ops) silos in IT environment through systems approach	Hussaini, S. W.	2014	IEEE	68
SLR 44	Toward unified DevOps model.	Wahaballa, Abubaker, et al.	2015	IEEE	69
SLR 45	Understanding DevOps & bridging the gap from continuous integration to continuous delivery	Virmani, M.	2015	IEEE	70
SLR 46	Analysis of Devops Tools using the Traditional Data Mining Techniques.	Vaasanthi, R., Kumari, V. P., & Kingston, S. P.	2017	Proquest	71
SLR 47	Encouraging knowledge contribution in IT support: social context and the differential effects of motivation type.	Wang, X., Clay, P. F., & Forsgren, N.	2015	Proquest	72
SLR 48	Automation of Continuous Services: What Companies of Latvia Says about It?.	Bartusevics, A.	2017	Science Direct	73
SLR 49	Bridging On-site Practices and Design Principles for Service Development	Hosono, S., & Shimomura, Y.	2017	Science Direct	74
SLR 50	Continuous deployment of software intensive products and services: A systematic mapping study	Rodríguez, P., Haghighatkhah, A., Lwakatare, L. E., Teppola, S., Suomalainen,	2017	Science Direct	75

		T., Eskeli, J., ... & Oivo, M.			
SLR 51	Fitzgerald, B., & Stol, K. J. (2017). Continuous software engineering: A roadmap and agenda. Journal of Systems and Software, 123, 176-189.	Fitzgerald, B., & Stol, K. J.	2017	Science Direct	76
SLR 52	A Qualitative Study of DevOps Usage in Practice.	Erich, F. M. A., Amrit, C., & Daneva, M.	2017	Wiley	77

APPENDIX 2. Culture related practices

No	Practices Related values	Additional details	paper/s
1.	In microservices development, teams and subteams should develop services and test them individually	The environment of the development should be feasible stateless and reproducible	[SLR 2]
2.	Developers should be highly technical and capable of developing a system by considering the operational aspects	Additionally, developers should be able to support the architectural changes on development and operational sides	[SLR 3] [SLR 4] [SLR 5] [SLR 19]
3.	Architecture team should consider quality attributes and reducing the feedback cycle	Also it should consider reusability, aggregating logs, isolating changes, and testability	[SLR4] [SLR 5]
4.	Establish a high level of independency among team members.	splitting out modules or components of the applications in the architecture into vertical layers and then letting each team to be responsible for their own module or component.	[SLR4]
5.	Isolating changes	isolate changes to minimize the impact of changes.	[SLR4] [SLR 32]
6.	The requirements of operations team should considered as part of requirements gathering and architecture design processes.	Operations stakeholders care about fast failure detection, prediction, diagnosis, and recovery.	[SLR4]
7.	Improve collaboration by co-locating teams and discuss,		[SLR 5]
8.	Rapid feedback	Shorter feedback loop at each stage in CDP enables teams and team members to partner in producing high quality software.	[SLR 5]
9.	Joint work and shared responsibility	Sharing responsibility among different team members will increase the speed and the	[SLR 5]

		frequency to the demanded level by DebOps	
10.	Empowering and engaging operations personnel	Shifting some ops team responsibilities into development team	[SLR 5]
11.	Keeping Operations team a ware of any changes/configurations/release		[SLR 18] [SLR 4]
12.	Management should be restructured.	Additionally, New skills should be acquired.	[SLR 22]
13.	Transparency between teams and customers.	Identifying bugs before customers and create an efficient to handle customers feedback	[SLR 25]
14.	Prioritization of work and systematic testing practices.		[SLR 25] [SLR 33]
15.	constant communication channels with customers		[SLR 25] [SLR 43]
16.	Advertising for DevOps roles correctly		[SLR 31]
17.	Small changes.	Doing all this change in tiny increments at warp speed within the framework of a central strategy enables extremely rapid innovation and learning,	[SLR 32]
18.	Collaborate closely with stakeholders to articulate intent and agreement on the risk-mitigation strategies		[SLR 33] [SLR 43]
19.	Engage the right stakeholders.	IT infrastructure and operations, the business operations, legal and compliance to meet business requirements associated with balancing velocity and risk.	[SLR 33]
20.	Improve time to production by better detection and correction		[SLR 33]

21.	Reward or recognize teams or groups as a whole	Avoid individuality	[SLR 33]
22.	Leadership and the DevOps team must treat failure as a learning experience.		[SLR 33]
23.	DevOps can be applied to very different delivery models but must be tailored to the environment and product architecture.	Due to different machines and set up environments and tools	[SLR 41]
24.	Enhance customer centricity	by bringing in customer focus in applications and customer management	[SLR 43]
25.	defining and deploying Program Management Processes	To enhance communication between dev and ops teams	[SLR 43]
26.	Central Dashboard for all the stakeholder	This will help to monitor the performance and make better decision	[SLR 43]
27.	Enhance DevOps synergy by defining and implementing a nimble CFT (cross functional team)	For releases into production	[SLR 43]

APPENDIX 3. Tools reported in the literature

No	Tool	Classification/Description	Details	Paper/s
1	GitLab	Version Control and repository	• web-based Git repository manager • Wiki • issue tracking • open source	[SLR 1] [SLR 46]
2	Jenkins	continuous integration	• Open source • Automation • Supports version control tools	[SLR 1] [SLR 11] [SLR 41] [SLR 46]
3	Jira	issue tracker	• bug tracking • Issue tracking • Project management functions	[SLR 1] [SLR 5] [SLR 21] [SLR 46]
4	SonarQube	source code Analysis and continuous SQA	• Open source • Continuous inspection of code quality.	[SLR 1] [SLR 46]
5	OpenStack	Open Source Cloud Computing Software	• Open-source • Cloud computing	[SLR 2]
6	Docker	Deployment, managing containerized software deployments		[SLR 2] [SLR 4] [SLR 46]
7	Wiki	Documentation and collaboration	Collaborative website	[SLR 5]
8	Chef	Configuration management, Continuous delivery	• Streamline the tasks of configuring • Automation	[SLR 11] [SLR 28] [SLR 41] [SLR 46]
9	Puppet	Configuration management, Continuous delivery	• Open Source	[SLR 11] [SLR 41] [SLR 46]

10	UrbanCode	Continuous delivery	• Build automation	[SLR 11]
11	uDeploy	deployments	• Automation deployment	[SLR 11] [SLR 46]
12	Stash	distributed version control management		[SLR 21]
13	Bamboo	Continuous Integration Server		[SLR 21] [SLR 41] [SLR 46]
14	HockeyApp	Delivery Server		[SLR 21]
15	SPACE4Cloud	Tool for System PerformAnce and Cost Evaluation on Cloud)	• Specification, assessment and optimisation of cloud applications	[SLR 27]
16	Juju	Deployment	• Open source • Application modelling tool • Facilitating deploying, configuring, scaling, integrating, and performing operational tasks on private cloud services	[SLR 28]
17	TeamCity	Continuous integration	• Build management • Continuous integration	[SLR 41] [SLR 46]
18	Ansible	Deployment-configuration management	• Open-source • Automates software provisioning, configuration management • Application deployment	[SLR 41] [SLR 46]
19	Loggly	Logging	• Cloud-based • Log management • Analytics	[SLR 41]
20	GreyLog	Logging	• Open Source	[SLR 41]
21	Nagios	Monitoring	• Open source	[SLR 41] [SLR 46]

			• monitors systems, networks and infrastructure.	
22	New Relic	Monitoring	• Digital performance monitoring	[SLR 41]
23	Cacti	Monitoring	• Open source • Web-based network monitoring	[SLR 41]
24	RUNDECK	Continuous integration		[SLR 46]
25	TRAVIS CI	Continuous integration	• Build • Test software projects hosted at GitHub.	[SLR 46]
26	Circle ci	Continuous integration	• Continuous integration • Delivery platform	[SLR 46]
27	SaltStack	Configuration Management	• Cloud management	[SLR 46]
28	CFengine	Configuration Management	• Open source • Automated Configuration • Maintenance	[SLR 46]
29	Windows Server IIS7	Deployment		[SLR 46]
30	Amazon	Deployment		[SLR 46]
31	Gradle	Build	• Open source • Build automation	[SLR 46] [SLR 41]
32	Maven	Build	• Build automation tool used primarily for Java projects.	[SLR 46]
33	Logstash	Monitoring	• Events and logs management	[SLR 46]
34	MONIT	Monitoring	• Open source	[SLR 46]
35	Splunk	Monitoring	• Log management • Monitor and Analyze • Visualize machine data.	[SLR 46]
36	AppPerfect	Monitoring		[SLR 46]

37	Nexus	Version Control and repository		[SLR 46]
38	Coverity	Code Analyzer	● Static code analysis ● Dynamic code analysis tools	[SLR 46]
39	Fortify	Code Analyzer	Analyze: ● Data flow, ● Control flow, ● Semantic, ● Structural, ● Configuration, ● Buffer.	[SLR 46]
40	JS hint	Code Analyzer	● Detect errors and potential in javascript	[SLR 46]
41	Blue optima	Code Analyzer		[SLR 46]
42	CheckStyle	Code Analyzer	● Java ● Automate code test	[SLR 46]
43	Gerrit	Code Analyzer	● Team code collaboration tool ● Team review ● Free ● Web based	[SLR 46]
44	Jacoco	Code Analyzer		[SLR 46]
45	Clover	Code Analyzer		[SLR 46]
46	Semmle	Code Analyzer		[SLR 46]
47	GIT	Version Control and repository	Tracking changes in computer files	[SLR 46]
48	Archiva	Version Control and repository		[SLR 46]
49	SharePoint	Version Control and repository		[SLR 46]
50	artifactory	Version Control and repository		[SLR 46]

APPENDIX 4. Tools reported by interviewees.

Category	Tools
Cloud Services:	• AWS • Google Cloud • Azure • Rackspace • OpenStack • Heroku
Continuous Delivery / Continuous Integration:	• Jenkins / Hudson • Bitbucket • TeamCity • Bamboo • Visual Studio
Deployment:	• Capistrano • CodeDeploy • Octopus
Code Versioning tools/Repositories:	• GitHub • Bitbucket • GitLab • SVN
Configuration Management Tools:	• Chef • Puppet • Ansible • SaltStack
Build/Automation Packaging:	• Maven • Gradle
Artifacts Repositories:	• Nexus

	• S3
Virtualization:	• Vagrant • Docker
Scripting:	• Linux Bash • Windows batch / powershell scripting
Security	• Vault
Code Checking:	• Sonar • checkStyle • findBugs
Testing:	• Junit • TestNG • Maven • Selenium • Jmeter
Monitoring:	• New relic • Dynatrace • Nagios • Zabbix
Logging:	• Splunk • Logstash • Log.io
Database	• DbMaestro • Liquibase • Datical