TÉCNICO LISBOA

# DevOps Maturity Model

**Yasser Rossano Manuel Zacarias**

Thesis to obtain the Master of Science Degree in

**Information Systems and Computer Engineering**

Supervisors: Prof. André Ferreira Ferrão Couto e Vasconcelos
Prof. Miguel Leitão Bignolas Mira da Silva

**Examination Committee**

Chairperson: Prof. Paolo Romano
Supervisor: Prof. André Ferreira Ferrão Couto e Vasconcelos
Member of the Committee: Prof. José Luís Brinquete Borbinha

**January 2021**

# Acknowledgments

This thesis is the conclusion of two and a half years of studying computer science and engineering MSc. at the Instituto Superior Técnico. It has been an amazing time of personal growth, both inside and outside of the Instituto environment. This thesis has taken more than 6 months to complete, however, I would not have been able to manage this without the help of the other people.

To my two mothers Luísa Manuel and Ana Madaleno, I am very thankful for granting me this desired opportunity and because for believe in myself when I thought I was not able.

I would like to acknowledge my dissertation supervisors Prof. André Vasconcelos and Prof. Miguel Mira da Silva for their insight, patience, support and sharing of knowledge that has made this Thesis possible.

To Curry Cabral Hospital medical team, I am thankful for everything they have done for me in the most difficult times. In particular, I would like to thank occupational Therapist Inês Osório for her professionalism, attention, patience, and immense sympathy.

I would like also to thank my parents for their friendship, encouragement and caring over all these years, for always being there for me through thick and thin and without whom this project would not be possible. I would also like to thank my aunts, uncles and cousins for their understanding and support throughout all these years.

Last but not least, to all my friends and colleagues that helped me grow as a person and were always there for me during the good and bad times in my life. Thank you.

# Abstract

The expanding pace of business competitiveness and environment dynamic imposes rapid changes to organizations, in many areas including Information Systems (IS), which is mainly responsible to ensure the structural division of software development and system operation. Although the team structural division, there is a need to maintain cohesion between them. DevOps is a collaborative and multidisciplinary effort in software development to bridge the gap between the Development and Operation teams. However, many organizations struggle with soft aspects of DevOps and also in breaking the barrier that can be created by other parts of the organization to get on board. Ideally for successful DevOps implementation collaboration and cooperation are needed, especially regarding management support.

The research methodology used throughout this research was Design Science Research. Additionally, to research method, Systematic Literature Review was performed in which the following results were obtained: DevOps concepts(32), processes (9), practices (33), and roles (20). as a result of this, a Process Reference Model (PRM) and Process Assessment Model (PAM) was developed and were the foundation for the maturity model. A maturity model was created grounded on PRM and PAM to provide a reliable tool for organizations to evaluate the maturity and provide guidance to achieve higher levels of DevOps maturity. The artefact was demonstrated and evaluated in one Organization. Therefore, the results point out that the proposed maturity model is a valuable instrument for the organization.

# Keywords

DevOps; Maturity Model; Process Assessment Model; Process Reference Model.

# Resumo

O ritmo crescente da competitividade das empresas e da dinâmica do ambiente impõe mudanças rápidas às organizações, em muitas áreas, incluindo os Sistemas de Informação (IS), que são os principais responsáveis por assegurar a divisão estrutural do desenvolvimento de software e do funcionamento do sistema. Embora a divisão estrutural da equipe seja necessária para manter a coesão entre eles. DevOps é um esforço colaborativo e multidisciplinar no desenvolvimento de software para preencher a lacuna entre as equipes de Desenvolvimento e Operação. No entanto, muitas organizações lutam com os aspectos brandos do DevOps e também para quebrar a barreira que pode ser criada por outras partes da organização para entrar no mercado. Idealmente, para que a implementação do DevOps seja bem-sucedida, é necessária colaboração e cooperação, especialmente no que se refere ao suporte de gerenciamento.

A metodologia de pesquisa utilizada ao longo dessa pesquisa foi o Design Science Research. Além disso, no que respeita ao método de investigação, foi efetuada uma revisão sistemática da literatura, na qual foram obtidos os seguintes resultados: Conceitos DevOps(32), processos (9), práticas (33) e funções (20). como resultado, foi desenvolvido um modelo de referência de processo (PRM) e um modelo de avaliação de processos (PAM), que foram a base do modelo de maturidade. Foi criado um modelo de maturidade baseado em PRM e PAM para fornecer uma ferramenta confiável para as organizações avaliarem a maturidade e fornecerem orientação para alcançar níveis mais altos de maturidade de DevOps. O artefato foi demonstrado e avaliado em uma Organização. Por conseguinte, os resultados indicam que o modelo de maturidade proposto é um instrumento valioso para a organização.

# Palavras Chave

DevOps; Maturity Model; Process Assessment Model; Process Reference Model;

# Contents

# List of Figures

# List of Tables

# Acronyms

| | |
|---|---|
| **DSRM** | Design Science Research Methodology |
| **SLR** | Systematic Literature Review |
| **PRM** | Process Reference Model |
| **PAM** | Process Assessment Model |
| **IS** | Information System |
| **MM** | Maturity Model |
| **SDLC** | Software Development Lifecycle |
| **CMMI** | Capability Maturity Model Integration |

# 1

# Introduction

## Contents

Nowadays, the business environment is very dynamic, which imposes rapids changes in many areas including Information System (IS), that mainly makes the structural division of software development and system operation [5] [6]. This often also represents a challenge to IS, because of the historic gap between Dev and Ops [7], characterized by the fact that Dev and Ops teams, each pursuing their own different objectives which are not efficient and can make communication, collaboration, and issue resolution almost impossible. Note that the lack of synergies can affect the communication and collaboration producing undesirable results.

For decades organizations have been looking for new ways to improve their software development processes to keep up with business and market demands [8] [9].In past decade DevOps, a new approach originated in the context of agile software development movement combining development and operations [10] [11], but focusing on development, quality assurance and operations aspects [12], allowed the integration of development and operations teams to achieve fast high-quality releases [13] [14] [15], while agile practices are mainly focused on rapid interactive development side aspects of IS and little attention is given to the operations. However, successful implementation of DevOps benefits greatly from taking agile approach of software development process.

Getting development and operations to collaborate and communicate effectively is imperative, regardless of whether the team is embracing agile or any other methodology [16]. It implies wider collaboration and communication between teams.

The approach to DevOps whose can be seen as a cultural movement that aligns people, process, and technology with a common objective of increasing value and eliminate waste using some associated technologies [15] has strongly developed over decades which is now widely adopted in software engineering companies.

This results in DevOps being an integral part of Software Development Lifecycle (SDLC). According to [17] [18] asserted that DevOps was “a conceptual framework which aims at befitting IS development by integrating development and operations in various ways”.

Maturity Model (MM) “offer organizations a simple but effective possibility to measure the quality of their processes” [19], and it has been used as a tool to assess the effectiveness of organizational processes, capabilities or business intelligence.

Although the benefits that DevOps can bring to development and operations, it might not be always successful. This approach is not a simple straightforward task and involves the consideration of a set of challenges such as organizational, communication patterns, process, and technical. According to [20] [21] [22] organizational challenges may refer to organizational culture, enterprise data models, IT operating models, reward models, and risk allocation. Also the assessment methods for the maturity models where the literature lacks detailed description, that prescribe how to assess the DevOps adoption process for organizations, so they can improve their maturity incrementally [23].

The document consists of creating a process reference and process assessment model which together form the DevOps maturity model based on ISO 3300xx family. Perform a Systematic Literature Review (SLR) represents the first step to create a process reference model which allow to identifying those processes that

are closely related to DevOps. After that, process assessment model allow to assess the capability maturity level of each process found by a Company and then provide criteria and characteristics that need to be fulfilled to reach a particular maturity level. Once the foundation of the maturity components are built and established, the maturity model will be tested by realizing a self-assessment to the company adopting DevOps, in order to validate the proposal.

This proposal represents a different approach to DevOps maturity model and it is structured following processes identified in the literature. Remember that the DevOps maturity model proposed is not to be considered as the final product, rather it should be viewed as a guide (and should) be improved and customized to match the actual needs of the company.

## 1.1 Research Problem

The DevOps phenomenon is gathering pace as more organizations seek to leverage the benefits it can potentially bring to software engineering functions. This approach helps deliver value faster and continuously, reducing problems due to miscommunication between team members and accelerating problem resolution. It raises the questions about the point of the time in which companies needs to analyze the techniques and approaches followed by them, regarding the implementation of DevOps, and assess where they are standing in terms of adoption, what capabilities they need to acquire, and what tools can support along this process in order to improve upon their performance.

MM has been used to assess adopted practices, processes, capabilities (Capability Maturity Model) or the business intelligence in the organization. A Maturity Model for DevOps could be a very useful metric to evaluate and analyze the tools, techniques, and approaches followed by an organization concerning DevOps.

Succinctly, the problem that we aim to address **is the lack of DevOps maturity model based on the ISO standard to guide companies to assess the maturity of their DevOps adoption, by helping them assess their current DevOps maturity and move it to a next maturity level.** This resulted in the following research objectives for this study:

***RO1.Propose a DevOps maturity model to assess the current state of DevOps adoption in an organization.***

***RO2.Provide guidance to achieve higher levels of DevOps maturity.***

In order to design an effective and comprehensive DevOps maturity model, it is necessary to research the design context. Thus, questions were raised to address the objective of this research.

***RQ1. What is a suitable DevOps Maturity framework for assessing the maturity of a DevOps environment in any organization?***

To address this problem we follow design research methodology, which is further detailed in chapter 2.

## 1.2 Organization of the document

The remainder of the document is organized as follow:

**Chapter** 1 **("Introduction")** presents the motivation, research problem and their motivations.

**Chapter** 2 **("Research Methodology")** describes the research methodology used to conducting and guide the research.

**Chapter** 3 **("Theoretical Background")** provides the background concepts for this research, beginning with DevOps in a general way, but focusing in several definitions of DevOps presented in the literature and then maturity model is discussed briefly, a brave approach to Capability Maturity Model Integration (CMMI), and finally, Process Reference Model (PRM) and Process Assessment Model (PAM) are discussed.

**Chapter** 4 **("Systematic Literature Review")** presents a systematic review about DevOps concepts, processes, practices and roles as well as how processes and practices are related and practices and roles as well. **Chapter** 5 **("DevOps Maturity Model Proposal")** describes PRM, PAM, and the proposed maturity model.

**Chapter** 6 **("Demonstration")** By means of an application test with one participant organization the applicability and usability of the artefact were demonstrated. The utility of the MM will be further validated by DevOps experts.

**Chapter** 7 **("Evaluation and Communication")** the artefact will be evaluated in terms of quality, utility and efficacy and communicate the problem, the importance, the utility, the rigour and the effectiveness of its design.

**Chapter** 8 **("Conclusion")** summarize and conclude the project, presenting the results obtained from the literature review and the maturity model as well as the limitations during the execution of the research and future work.

# 2

# Research Methodology

## Contents

For conducting and guide this research the Design Science Research Methodology (DSRM), provides a process model for doing research in Information Systems and other applied resource disciplines, as well as a mental model for reviewers to evaluate researchers. [16].

## 2.1 Design Science Research

**Problem and motivation identification** – In this section, we identify the importance and purpose of the research problem. Therefore, we identified one problem that motivated this literature. From the problem statement in chapter 1, it became clear a need exist for a DevOps maturity models and frameworks, as well as more empirical studies addressing the subject DevOps in general.

**Define the objectives for a solution** – A maturity model will be created (chapter 5) to address the identified problem. The solution will consist of a literature review and experts validation of the maturity model.

**Design and development** – The design of the model consists of three steps. First, identification of the DevOps processes by the literature review (Chapter 4). Secondly, the process reference model is created based on the identified DevOps processes. Finally, process assessment is created according to the existent PRM as it is described in chapter 5.

**Demonstration** - The newly created DevOps maturity model is used for self-assessment in Company A, for measuring the maturity of the respective DevOps implementation. This will provide insights into the applicability of the model (Chapter 6).

**Evaluation** - Observe and measure how well an artefact supports a solution to the problem, comparing the objectives to the results observed from the use of the artefact in the demonstration(Chapter 7).

**Communication** - The results of this study is communicated in two manners; this research report and the article derived from them (Chapter 8).

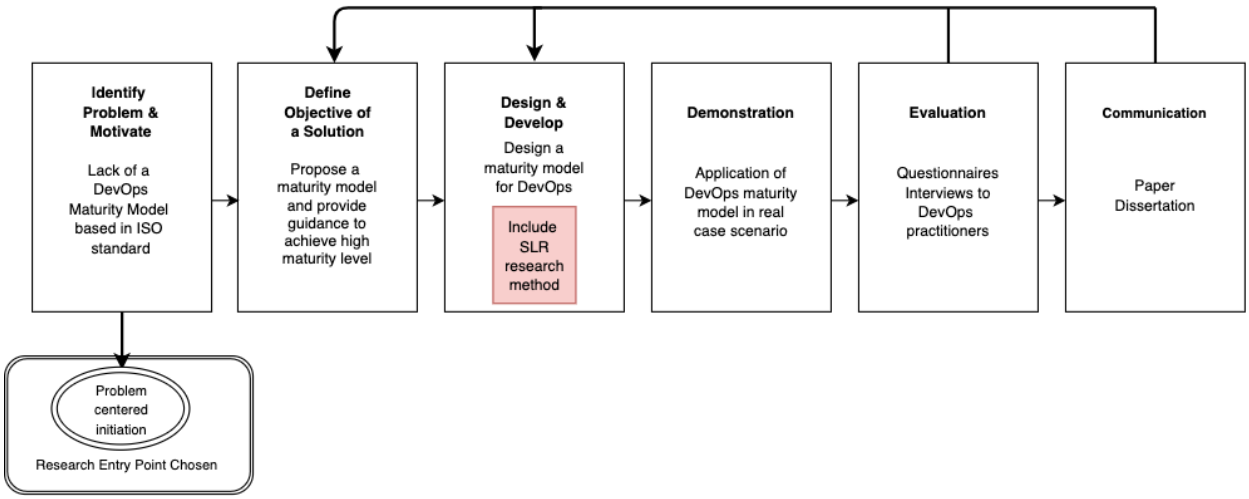


**Figure 2.1:** Research method phases and activities [1]

Additionally, to research method, in the design and development phase, Systematic Literature Review (SLR), was incorporated with the goal of achieving a higher reliable and consistent result when trying to identify processes related with DevOps. The section below describes the method and the associated procedures step by step.

## 2.2 Systematic Literature Review

A systematic literature review involves a variety of procedures that aim to ensure comprehensive and accurate analysis in order to obtain valid knowledge through empirical studies in a given area. According to [2] definition "a systematic 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". In addition, a systematic review of the literature allows us to identify the current literature, its limitations, quality and potential. In contrast to the simple and conventional review of the literature, a systematic literature review presents differentiating features such as [2]:

- Systematic reviews start by defining a review protocol that specifies the research question being addressed and the methods that will be used to perform the review.
- Systematic reviews are based on a defined search strategy that aims to detect as much of the relevant literature.
- Systematic reviews document their search strategy so that readers can access its rigour and completeness.
- Systematic reviews require explicit inclusion and exclusion criteria to assess each potential primary study.
- Systematic reviews specify the information to be obtained form each primary study including quality criteria by which to evaluate each primary study.
- A systematic review is a prerequisite for quantitative meta-analysis.

A structured literature review, implementing the practices of Kitchenham [2], comprises three consecutive stages and they must be executed in the following order: planning, conducting, and reporting.

The choice of SLR as the research methodology is based on the purpose to identify the processes, concepts, practices, and roles of DevOps. Since the processes constitute the foundation for a Process Reference Model (PRM). The conducted SLR is described in Chapter 4.

# 3

# Theoretical Background

## Contents

The description of concepts and definitions, associated with our topic and derived from existing theories and studies available in the academic literature, is provided in this section. The literature review can be regarded as an important part of the theoretical background since it involves applicable and relevant academic work.

## 3.1 DevOps

The first usage of DevOps, a combination of development and operation, stems from a presentation during the 2008 Agile conference by Debois and Shafers [24]. Even though this is more than a decade ago it seems that research on DevOps is still in its infancy [25]. However, the DevOps movement has been widely discussed for nearly a decade, it still lacks a widely endorsed definition. This section will present how some studies tried to define DevOps.

"DevOps is a collaborative and multidisciplinary effort within an organization to automate continuous delivery of new software versions while guaranteeing their correctness and reliability." [26]

"A set of practices intended to reduce the time between committing a change to a system and the change being placed into normal production while ensuring high quality" [27]

"DevOps is a set of practices that is trying to bridge developer-operations gap at the core of things and at the same time covers all the aspects which help in speedy, optimized and high-quality software delivery" [22]

Recent studies describe DevOps as vaguely defined and loosely used in the software engineering community [8]. DevOps has been described, among other things, as a movement, philosophy, a practice or a culture. However, from the literature, it becomes clear that DevOps aims to decrease the time to production environments for changes made to the software [14] by facilitating a lean connection between development, delivery, and operation [14].

Conforming to Humble and Molesky achieving both frequent, reliable deployments and a stable production environment with DevOps is not a zero-sum game. Therefore, it can be achieved through four main principles: culture, automation, measurement and sharing [27]. Lwakatare et al. also mentioned these principles and adds monitoring as well as collaboration and communication [24]. But sharing is also mentioned as part of collaboration and culture, in order to prevent redundancy, it was excluded of these five dimensions, that will be shortly addressed in the following section.

## 3.2 Collaboration and Communication

The basic DevOps aspect of collaboration is the implied collaboration between the software development and Operations teams, promoted by their practices and principles that aim to break down functional and physical separation among teams.Thus, in this collaborative environment, team are encouraged and feel technically qualified to operate in different areas as a single unit [28]. Ghantous and Gill literature research of 2017, researched this

more deeply and showed that collaboration and communication is the most frequently mentioned conceptual element to describe DevOps reaching 23 percents in overall of articles [22]. Lwakatare splits this up in two main practices [24]: “Increasing the scope of responsibilities” and “intensifying cooperation and involvement in each other daily work.”

## 3.3 Automation

Automation underlines most of the practices that constitute DevOps. According to the assumption made by Humble in regard to DevOps: “achieving both frequent, reliable deployments and a stable production environment” [24]. This can be achieved by creating a continuous delivery process which consists of continuous planning, integration, deployment, testing, and monitoring [29]. For example, automatic deployment is the ability to quickly deploy new releases into a production environment [27]. This automation can be supported by using an IT setup named infrastructure as code (IaC) [24].

The IaC promotes managing knowledge and experience of the plethora of subsystems as a single commonly available source of truth [30], its concept is based on the idea that the entire infrastructure provisioning should be maintained as code in a source code repository [22]. This means that the infrastructure is only maintained by executable code which can create the same environment repeatably. This can prevent issues from lacking documentation, mistakes made between environments and transitioning of employees [31].

## 3.4 Culture

The Changeover to a new culture can be difficult, however, Shamow described the focus on the change of culture in companies is a necessity for adopting DevOps [32]. Some of those changes, as described by Shamow, are the importance for people inside the company to know the seriousness of bypassing the DevOps teams in crises, to not worry about specific tools and to provide full transparency between groups.

Lwakatare et al. describes the necessary DevOps culture as empathic, supportive and having good working environments between development and operations [24]. An important step forward is to be involved in each other’s work. This can be accomplished by rotating Developers inside the operation teams, both teams should have regular meetings with the other team and both should be responsible for production, which implies also that developers should be on call for production issues [27].

To create this culture, it is important to hire the people that have the right knowledge of automation, by comparing the knowledge, skills, and abilities of future employees to the capabilities needed for DevOps [33]. Organizations should give employees trust in making the right and necessary decisions for implementing automation of the process by creating a culture of personal responsibility [32].

## 3.5 Monitoring

Monitoring is multipurpose, it can be used by developers to make sure that the deploying software is performing correctly and give us a thorough understanding of the current health of the system, allowing to detect problems early on, react to them in a timely fashion, and to prevent problems from arising. For example, by monitoring the physical capabilities of the system (CPU, memory, hard disk space) with effective tools, this can prevent system crashes or applications getting too slow by adding enough resources before those issues occur. Also, monitoring can help developers to quickly recover from code failures or evaluate the code coverage for automatic tests [24].

## 3.6 Measurement

Measurement is very important for evaluating the success of both the development and operation teams. This can be achieved by monitoring high-level and low-level metrics. For instance, high-level business metrics can be the total revenue or end-to-end transactions per unit time. For the lower-end, it is important to use similar metrics for both teams as these key performance indicators influence people's behaviour, for example, the impact of product releases on the stability of the affected systems [27]. Using similar metrics ensures that both teams are trying to achieve the same goal, a stable production environment.

## 3.7 Maturity Models

In the past decades' maturity assessments of processes have become a well-established practice as a successful means for improving software-intensive organizations [11] [34]. A maturity model is a conceptual model that consists of a sequence of discrete maturity levels for a class of processes in one or more business domains. It represents an anticipated, desire or typical evolutionary path for these processes. They are commonly used as an instrument to conceptualize and measure the maturity of an organization or a process regarding some specific target state [35]. by the other hand, perceived benefits of maturity models resulted in the creation of a wide range of software process capability and maturity models.

In regard to maturity models, they normally can be categorized as fixed maturity level of focus area maturity model [41]. The fixed maturity level model distinguish a fixed number of maturity level, in which each maturity level is associated with a number of processes that have to be implemented [42], often following maturity levels as defined in Capability Maturity Model (CMM) [42]: 1. Initial, 2. Repeatable, 3. Defined, 4. Managed and 5. Optimized, with in each maturity level a number of Key Process Areas (KPA). The fixed maturity level models can be divided into staged maturity models, in which all KPA need to be in place to achieve a certain maturity level, and continuous maturity models, allowing for more gradual a varying improvement path by allowing KPAs to be scored at different levels [11] [43]. by the other hand another type of maturity model consists of focus

**Table 3.1:** Fundamental Maturity Models Concepts adapted from [4]

| Elements | Description |
|---|---|
| Dimension | Dimensions are domains or categories for capabilities, i.e. a set of related capabilities. It is recommended to formulate dimensions in exhaustive and mutually exclusive manner [4] |
| Capability | At the very core of maturity models are capabilities that are related to objects such as project or knowledge management [36] [37] [38] [39] |
| Level | Levels represent archetypal stages of maturity. Each level is related to a specific set of capabilities that ultimately should be empirically testable [40] |
| Core model | The core (maturity) model represents the relationships between dimensions, capabilities, and levels [40]. |
| Assessment instrument | The assessment instrument is based upon the core model assigning testable assessment criteria to each of the dimensions and levels [40]. |

areas that define levels of capabilities in various functional areas. Each focus area has a series of development steps for progressively mature capabilities and each focus area can have its own maturity level scale [42].

According to [44] application of maturity models can be supported by predetermined procedures. Roughly it consists of multiple iterations of defining the problem and relevance, comparing existing maturity models, maturity model development, followed by, gathering empirical evidence by evaluation in real-life situations.

The Capability Maturity Model Integration (CMMI) model is an exception to most maturity models, as much empirical evidence on this model exists and the CMMI has been used as a framework for many other models. The value of applying maturity models, more specifically SW-CMM (predecessor of CMMI) and CMMI, in software development and maintenance environment, has been shown by multiple researchers through empirical research [45] [46]. The CMMI-based process improvement resulted in better project performance and higher quality products through cost reduction, better scheduling of requirements, better quality products, higher customer satisfaction and higher return of investment as described by Goldenson and Gibson based on their research in 35 organizations [47] [48].

## 3.8 Process Reference Model (PRM)

When it comes to introducing the concept of Process Reference Model (PRM), it is pertinent to also cover the clarification of reference model. Regarding its explanation, “a reference model is an abstract framework for understanding significant relationships among the entities of some environment that enables the development of specific architectures using consistent standards or specifications supporting that environment. A reference model consists of a minimal set of unifying concepts, axioms and relationships within a particular problem domain, and is independent of specific standards, technologies, implementations, or other concrete details.” [49].

Each process in the PRM has the following descriptive elements:

- Process ID: Each process belonging to a Group is identified with a Process Identifier [ID] consisting of the Group abbreviated name and a sequential number of the process in that Group.
- Name: The name of a process is a short phrase that summarizes the scope of the process, identifying the principal concern of the process, and distinguishes it from other processes within the scope of the process reference model.
- Context: For each process, a brief overview describes the intended context of the application of the process.
- Purpose: The purpose of the process is a high-level, overall goal for performing the process.
- Outcomes: An outcome is an observable result of the successful achievement of the process purpose. Outcomes are measurable, tangible, technical or business results that are achieved by a process. Outcomes are observable and assessable.

## 3.9 Process Assessment Model (PAM)

Maturity models have been growing in considerable numbers if judged by the number of publications by academics and practitioners [50] [51]. The business world has also adopted maturity models to improve its business processes considering the quality management required by stakeholders and for reasons of competitiveness.

According to CMM's first version, "the CMM was designed to guide software organizations in selecting process improvement strategies by determining current process maturity and identifying the few issues most critical to software quality and process improvement."

Regarding standards, ISO/IEC 15504 was the first consensual standard that proposed a reference model for maturity models. The associated and updated standard now for ISO/IEC 15504 is the ISO/IEC 330xx family.

With the objective of performing an assessment, the document ISO/IEC 33002 defines the minimum set of requirements needed to achieve objectives, and consistent results, to form a structure for the assessment of process and the application of process assessment

- facilitates self-assessment;
- provides a basis for use in process improvement and capability determination;
- produces a process rating;
- addresses the ability of the process to achieve its purpose;
- is applicable across all application domains and sizes of organization;
- can provide an objective benchmark between organizations.

The ISO/IEC 33072 is characterized by a two-dimensional structure. A process dimension, the processes are defined and more specifically the PAM expands the PRM process definitions by including a set of process performance indicators called base practices for each process. The PAM also defines a second set of indicators of process performance by associating inputs and outputs in each process.

A capability dimension, the capability level and process attributes are used and expanded through the inclusion of a set of generic practices. Hence, a set of process attributes grouped into capability levels are defined. The processes attributes provide the measurable characteristics of the process quality characteristic of process capabilities.

The capability levels are defined on a six-point ordinal scale in a range of 0 to 5 with the respective order [49].

- Process capability Level 0: Incomplete process
- Process capability Level 1: Performed process

  1.1 Process performance process attribute
- Process capability Level 2: Managed process

  2.1 Performance management process attribute

  2.2 Work product management process attribute
- Process capability Level 3: Established process

  3.1 Process definition process attribute

  3.2 Process deployment process attribute
- Process capability Level 4: Predictable process

  4.1 Quantitative analysis process attribute

  4.2 Quantitative control process attribute
- Process capability Level 5: Innovating process

  5.1 Process innovation process attribute

  5.2 Process innovation implementation process attribute

,

# 4

# Systematic Literature Review

## Contents

As described in section Research Methodology and according to DSRM develop and design phase, it was chosen to start a literature review on the existent DevOps concepts, practices, processes, and roles to create a comprehensive overview of the context of DevOps processes.

## 4.1 Literature Review Method

This section aims to present the SLR approach [2] [52], by following the proposed guidelines to identify, analyze, and interpret all available and relevant literature published in the context of DevOps concepts, processes, practices, and roles. The SLR method comprises three consecutive stages: planning, conducting, and reporting. Figure 1 illustrates the phases and activities comprised by the SLR.

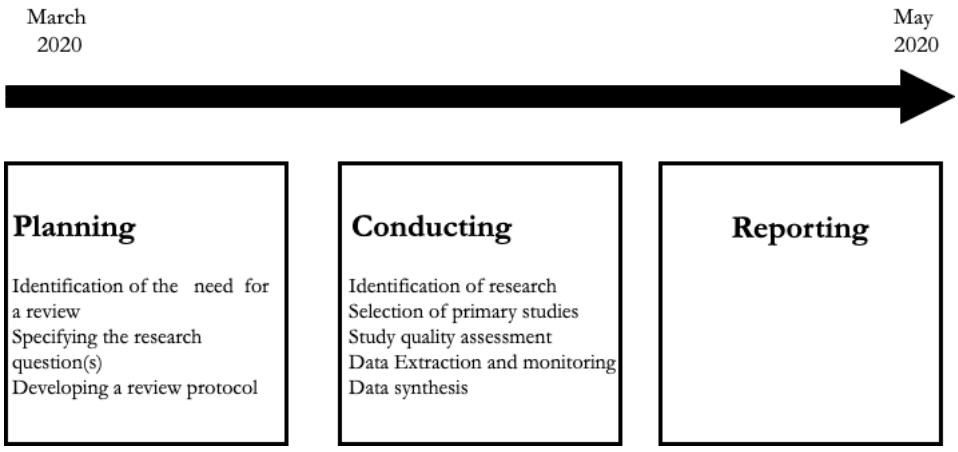


**Figure 4.1:** Research method phases and activities [2]

To manage and document the SLR, the software Mendeley[1] was used to support the process.

In the initial phase, planning review, the first step consists of identifying the need for a review. This section focuses on the remaining planning phase, which comprises the definition of the research questions and the development of the review protocol, which outlines the procedures for the conducting phase that includes the search process; the establishment of inclusion and exclusion criteria; quality assessment instrument, and the data extraction and synthesis strategy. Furthermore, the conducting and reporting phases are described in section 4.6.

## 4.2 Research Questions

The research work intends to answer the following research questions:

- RQ1: What are the DevOps concepts?

[1]Mendeley: https://www.mendeley.com/.

- RQ2: What are the DevOps processes?
- RQ3: What are the DevOps practices?
- RQ4: What are the DevOps roles?

To address RQ1, we explore the studies explicitly mentioning the related concepts to DevOps.

With RQ2, we aim to describe the key process of DevOps, as well as understand the related stages and phases.

Regarding RQ3, we aim to explore the practices, patterns, and strategies to enhance DevOps.

Finally, with RQ4 we aim to delve into the roles described in the studies, understand whether they can be related to practices or not, and if so, how they are related.

## 4.3 Search Process

The review comprises the following well-known academic databases:

- ACM Library
- AIS Library
- IEEEXplore
- Google Scholar
- Research Gate
- Semantic Scholar
- EBSCO.

The search conducted aims to find all literature, having a focus on academic studies. The papers returned were obtained by applying the search string to the title and abstract in each digital library. To increase confidence and reliability to the search on the databases, the process of search in the above mention digital libraries was done twice with a gap of one week. Table 4.1 shows the search string. The Boolean “OR” was used to join each part, whilst the Boolean “AND” was used to join major part.

**Table 4.1:** Search string

| **Scope** | **String** |
|---|---|
| DevOps Concepts | ”devops” AND (”standard” OR model OR ”maturity model”) |
| DevOps Processes | ”devops Framework”, ”devops methodology”, ”devops models” |
| DevOps Roles | devops roles |

To guide the research and identify the related literature that could address the research questions and add value to the study a manual search process, according to the minimum quality criteria to obtain the studies in consonance with the key elements stated in the research questions, was conducted. Further, this review included those studies that were published in English between 2007 and 2019. Within the selection of those years, was possible to cover the recent literature as well as having a balance of different approaches and how they evolved through the years.

After retrieved, papers were filtered considering the inclusion and exclusion criteria presented in section 4.4 the remaining were filtered first on title and abstract, and then on full text.

**Table 4.2:** Search configuration for each digital library

| Digital Library | Configuration |
|---|---|
| ACM Digital Library | Select items: The ACM Full-Text Collection<br>Search Within: Name (add generic search string)<br>Search Within: Abstract (add generic search string) |
| AIS Electronic Library | Select “Title” and add the generic search string<br>Click “+”, select “OR” and “Abstract”, and add the generic search string<br>Date range: Limit search: AIS Electronic Library (AISel) |
| IEEE Xplore | Query:”devops” AND (”standard” OR model OR ”maturity model”)<br>AND ”devops Framework”, ”devops methodology”, ”devops models”<br>Year: 2008-2020 |

## 4.4 Inclusion and Exclusion criteria

For this section, the inclusion and exclusion criteria were defined to filter the literature and identify the most relevant ones for this work.

To reduce subjectivity, this research work focuses on research that explicitly reports DevOps in the context of concepts, processes, practices, and roles. Based on this, the following inclusion criteria were developed:

- Article not including key elements of search string
- Empirical study included
- Clearly describes DevOps and its related concepts
- Focusses on the processes, practices, and roles

Criteria 1 was included to assure the reported paper attend the specific context of this SLR. Criteria 2 was selected with the aim of increase the scope of the research, allowing us to gather more papers. Criteria 3 and 4 were chosen to assess the key elements described in the research questions.

Furthermore, to ensure the quality and reliability of the literature studied, we excluded papers following criteria below presented:

- Article not published in English or Portuguese.
- Duplicate document (including the same papers published in a different digital library)
- Full text not available.
- Not including key elements of search string

## 4.5 Quality Assessment

For each selected paper a quality assessment was conducted to appraise the relevance and quality of their content.

The quality assessment of the selected articles was based on the four questions defined in our research. The scoring procedure was based on [2], where each question could have the possible answers: Yes (Y) =1 if the article answers to the questions with a 100 of certainty, partially (P) = 0.5 if the article does not fully answer the questions, and No (N) = 0 if the does not answer to any questions.

## 4.6 Data Extraction and Synthesis

The data extraction process was executed, according to the selection criteria above defined and complying with the objectives of this study, which were respectively; identify concepts, processes, practices, and roles in the context of DevOps and explore reflections of other methods.

After data extraction, it became evident that the studies adopted different approaches to address the same subject, although similarities were identified which helped to establish relations of some studies. From the titles and abstract it was not possible to deduce what was detailed in the studies, for this reason, 63 studies needed to be read in full text. Of these, only 30 studies were included. Figure 2 depicts the number of studies used in our research work following each specific research question.

## 4.7 Findings

This section describes the results obtained by conducting the SLR. The selection process was conducted based on a set of steps described in figure 4.2.

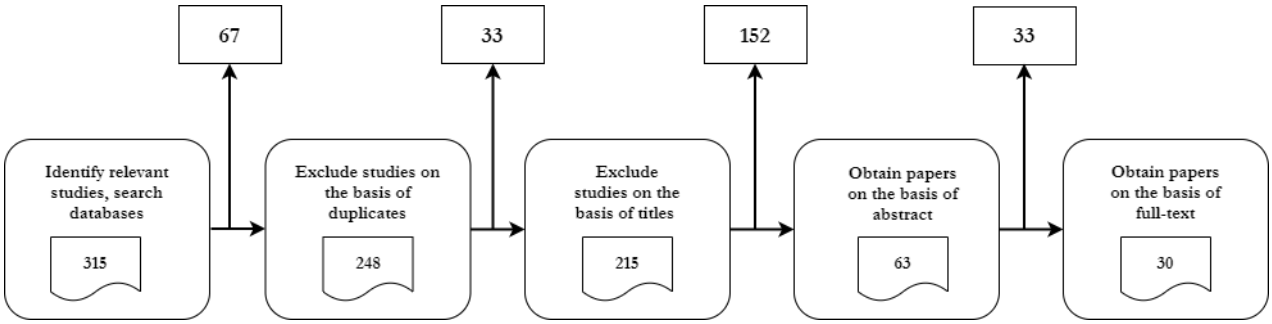


**Figure 4.2:** Paper selection process

The identified keywords and search terms mentioned in section 4.3 enabled to identify the relevant literature; however, it is important to recognize that the choice of keywords and search strings may have omitted relevant studies.

The search strategy applied at stage 1 of the review resulted in 315 studies including the possible set of duplicates. At stage 2, duplicates studies were removed automatically and then reviewed manually to ensure that no duplicates were left behind, allowing to reduce the number of studies to 248. At stage 3, reviews were made to exclude studies based on the titles, resulting in 215. At stage 4, reviews of all 215 abstracts and as result of this process 63 studies were selected. At the five and final stages, 63 studies were evaluated in detail focusing on the full text, some of the studies were excluded ending up with 30 primary studies as shown in appendix B.

As it is possible to notice in Figure 4.3, most of the selected papers were from ACM Digital Library, IEEE Xplore, Research Gate, followed by Springer Link. After that, AIS Electronic Library, Science Direct and Semantic Scholar with the same number of papers respectively.

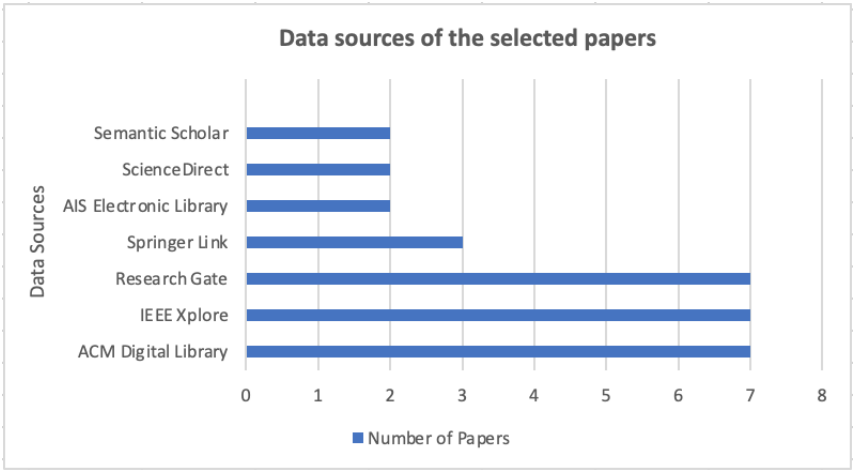


**Figure 4.3:** Data sources of the selected papers

From Figure 4.3 is possible to observe the distribution of the papers selected over the years and it shows that 2017 is the year from which more papers were selected for this research, this might indicate that in this particular year there was an increase of research in the field of DevOps.

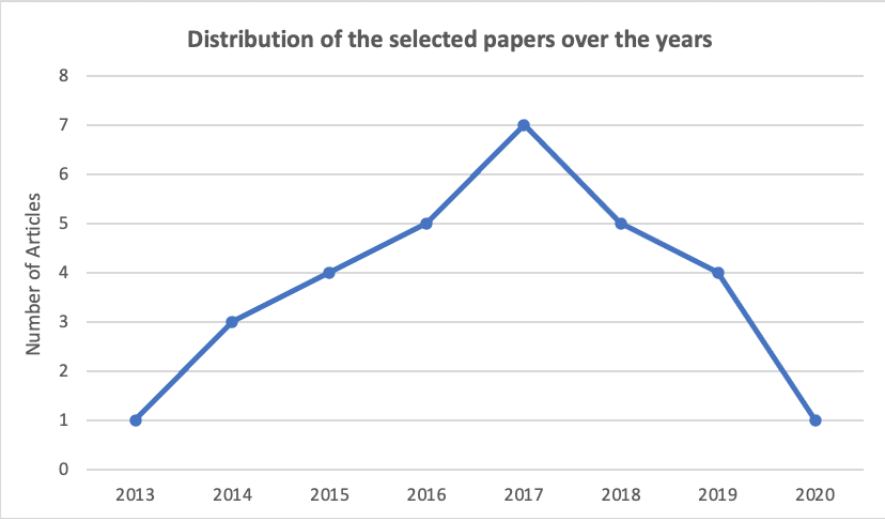


**Figure 4.4:** Distribution of the selected papers over the years

## 4.8 Quality Assessment

As discussed in section 4.5, the papers included in this literature were evaluated based on the quality assessment questions. Quality measures were applied to the selected studies to ensure that they are a valuable contribution to this SLR.

The score of each study is shown in table 4.3.

**Table 4.3:** Quality assessment of selected studies

| | QA1 | QA2 | QA3 | QA4 | TOTAL SCORE |
|---|---|---|---|---|---|
| [53] | 0 | 0 | 1 | 0 | 1 |
| [54] | 1 | 0 | 0 | 1 | 2 |
| [55] | 1 | 0 | 0 | 1 | 2 |
| [26] | 1 | 0 | 0 | 0 | 1 |
| [10] | 1 | 0 | 1 | 0 | 2 |
| [12] | 0 | 1 | 1 | 0 | 2 |
| [56] | 0 | 1 | 0 | 0 | 1 |
| [57] | 0 | 1 | 0 | 1 | 2 |
| [58] | 0 | 1 | 0 | 1 | 2 |
| [59] | 0 | 0 | 1 | 0 | 1 |
| [60] | 1 | 0 | 0 | 0 | 1 |
| [61] | 0 | 0 | 1 | 0 | 1 |
| [24] | 0 | 0 | 1 | 1 | 2 |
| [62] | 0 | 0 | 1 | 0 | 1 |
| [6] | 0 | 0 | 1 | 0 | 1 |
| [63] | 0 | 0 | 1 | 0 | 1 |
| [64] | 0 | 0 | 1 | 0 | 1 |
| [65] | 0 | 1 | 0 | 1 | 2 |
| [66] | 0 | 1 | 0 | 0 | 1 |
| [67] | 1 | 0 | 0 | 0 | 1 |
| [68] | 1 | 0 | 0 | 0 | 1 |
| [69] | 1 | 0 | 0 | 0 | 1 |
| [70] | 1 | 0 | 0 | 0 | 1 |
| [71] | 0 | 0 | 1 | 0 | 1 |
| [72] | 0 | 0 | 1 | 0 | 1 |
| [73] | 0 | 0 | 1 | 0 | 1 |
| [74] | 0 | 0 | 1 | 0 | 1 |
| [75] | 0 | 0 | 1 | 0 | 1 |
| [76] | 0 | 0 | 1 | 0 | 1 |
| [77] | 0 | 0 | 0 | 1 | 1 |

The results show that most of the papers proposed DevOps practices, being the most discussed subject (QA3, n=13), DevOps concepts were also highlighted with a certain emphasis since some papers clearly mention it (QA1, n=9). Moreover, we faced difficulties to classify papers as “Partially” in this quality assessment questions, because most of the studied literature meet one the questions, not mentioning elements of other question. Although, it is possible to highlight S1, S2, that addressed clearly more than one question, but do not provide further details related to other questions.

Also, worth noting that DevOps roles were discussed in less than half of the papers (QA2, n=7). Besides, only six papers reflect upon DevOps processes (QA2, n=6). It shows clearly the necessity of more research addressing and discussing these subjects.

The limitations of the presented results are traceable to the fact that the novelty of the subject had a strong

impact on the presented results. Although, structuring sub-questions could have helped increase the scope of research and possibly better results. Nevertheless, this study could provide new perspectives and ensure that important points are not overlooked in further researches.

Finally, the result of this review reported: DevOps Concepts, DevOps Processes, DevOps Practices, and DevOps Roles.

## 4.9 Concepts

For the concepts, studies explicitly using the term DevOps concepts, are shown in table 4.4, which presents a set of concepts and establish the comparison of the often-mentioned concepts in the studied literature. Therefore, we can highlight Continuous Delivery, Automation, Automated Pipeline, Communication and Collaboration, and Knowledge Sharing as most mentioned.

Although the presented literature approached DevOps differently, the concepts defined have similarities. Worth mentions that, some names descriptions of the concepts differed slightly and therefore have been grouped under the most descriptive name.

**Table 4.4:** DevOps concepts

| **Concepts** | [54] | [55] | [26] | [10] | [67] | [68] | [70] |
|---|---|---|---|---|---|---|---|
| Continuous Delivery | | | x | x | x | | x |
| Automation | x | x | x | | | | |
| Automated Pipeline | | | | x | x | | x |
| Communication and Collaboration | | | | x | x | x | |
| Knowledge Sharing | x | x | x | | | | |
| Continuous Integration | x | | x | | | | |
| Infrastructure as a code | x | | x | | | | |
| Process Sharing | x | x | | | | | |
| Shared Responsibilities | x | x | | | | | |
| Quality Assurance | x | | | x | | | |
| Activity Sharing | x | | | | | | |
| Artefact Management | | | x | | | | |
| Breaking down silos | | | x | | | | |
| Configuration Management | | | x | | | | |
| Containerization | | | x | | | | |
| Continuous Deployment | | | | x | | | |
| Continuous Feedback | | | | x | | | |
| Continuous Measurement | x | | | | | | |
| Continuous Planning | | | | x | | | |
| Continuous Runtime Monitoring | | | x | | | | |
| Culture of Collaboration | | | x | | | | |
| Deployment Pipeline | | | x | | | | |
| Log Management | | | x | | | | |
| Product Thinking | | x | | | | | |
| Release Engineering | | | x | | | | |
| Rollback Code | | | | x | | | |
| Shared Pipelines | x | | | | | | |
| Software Development Empowerment | | x | | | | | |
| Versioning | | | x | | | | |
| Virtualization | | | x | | | | |

## 4.10 Processes

The identified processes present a vast scope of what could involve a DevOps adoption. When analyzing the identified processes, it was possible to perceive that there is no standard definition for DevOps process according to [56], but a few different versions and implementations have been provided by the literature, in which the process is composed of stages that encompass software development and operation.

Table 5 shows the number of studies that mention the identified process (stages), although the lack of detailed information about process tasks. Worth noting that the literature does not suggest a special emphasis on a specific process (stages) but the process as Plan, Test, Release, Deploy, and Monitor stand out for the number of times they were cited. Therefore, the literature merely accentuates the need for the implementation of every process and its associated practices.

**Table 4.6:** DevOps process (phases)

| Process (stages) | References |
|---|---|
| Plan | [12] [56] [57] [58] [65] |
| Code | [12] [56] [65] |
| Build | [12] [56] [65] |
| Test | [12] [56] [57] [58] [65] |
| Release | [12] [56] [57] [58] [65] |
| Deploy | [12] [56] [57] [58] [65] |
| Operate | [12] [56] [65] |
| Monitor | [56] [57] [58] [65] |
| Optimize | [57] [58] |

## 4.11 Practices

To analyze the practices proposed as DevOps practices in the literature we attempted to identify those which have been explicitly presented as DevOps practices. Which is possible to highlight Continuous Delivery, Continuous Integration, Continuous Deployment, Continuous Monitoring, and Continuous Testing as the most cited. The table in appendix A shows the practices suggested in the primary studies.

## 4.12 Roles

From the carefully selected and reviewed literature, it was possible to identify DevOps related roles which are performed by different people in DevOps process (stages), however, it does not mean that they are all roles covering this subject. Therefore, it is important to highlight that roles vary according to the organizational structures.

Some of the roles names were correspondents, but to prevent repetition and possible redundancy they have been grouped under the most identifying name. Worth mention from table 4.7, the role of DevOps Engineer, which according to [57]"is emerging to facilitate the integrated process of DevOps to smoothly deploy software and hardware systems increments into production environment". [78] also added that DevOps Engineer was conceived with a mission to streamline the handoffs from development and operations. This illustrates that the establishment of DevOps requires new roles, in extension to existing traditional development and operation roles.

**Table 4.7:** DevOps roles

| **Roles** | [57] | [58] | [54] | [24] | [55] | [77] | [65] |
|---|---|---|---|---|---|---|---|
| DevOps Engineer | x | | x | | x | x | x |
| Network Administrator | x | | x | | x | | x |
| Developer | | | x | x | x | | |
| IT Manager | | x | x | | x | | |
| Tester | x | x | | | | | x |
| Quality Assurance Engineer | x | | | | | x | x |
| Cloud Engineer | | | x | | x | | |
| Database Administrator | x | | | | | | x |
| DevOps Consultant | | | x | | x | | |
| DevOps Supervisor | | | x | | x | | |
| Infrastructure and Management Team | x | | | | | | x |
| Orchestrator | x | | | | | | x |
| Security Engineer | x | | | | | | x |
| Support Analyst | | | x | | x | | |
| System Administrator | x | | | | | | x |
| System Engineer | | | x | | x | | |
| Technology Manager | | | x | | x | | |
| Application Architect | | x | | | | | |
| Developer and Operations | x | | | | | | |
| IT Professional | | x | | | | | |
| Product Owner | | x | | | | | |
| Project Manager | | | | x | | | |
| Release Manager | | x | | | | | |

## 4.13 Discussion

This section aims to discuss the answers to the defined research questions.

### 4.13.1 RQ1: What are the DevOps core concepts?

As the SLR study on DevOps described in section 2, we found that some studies are attempting to describe the most cited concepts in the literature. In fact, addressing all DevOps concepts in a single study would be very complex and challenging, due to the novelty of the research field and the quantity of the studies addressing specifically this subject.

In table 4.4 we identified concepts explicitly described in the literature and summarized them according to the number of times it was mentioned in the data sources, in which stands out automation, knowledge sharing, continuous delivery, infrastructure as a code, and process sharing. However, that does not necessarily mean that the table is complete or even that the order of the concepts presented in the table is the standard, worth noting that [10] suggested a different order for the concepts, although there are similarities between them.

On another hand, regarding concepts, [54] approached this subject differently by creating a division in which the DevOps concepts were grouped according to a category, like Core category, Enablers, Enablers and Outcomes, and DevOps adoption.

### 4.13.2 RQ2: What are the key DevOps processes?

The proposed DevOps processes are often supported by practices. [57] Proposed two key aspects focusing on continuous delivery pipeline and continuous improvement, which they considered as the most important aspects of the overall integrated DevOps lifecycle. However, [56] had a different approach, emphasizing that a DevOps process is normally composed by different stages and phases: plan, code, build, test, release, deploy, operate, and monitor; that can be periodically repeated for proper development and operation of the software.

Furthermore, a DevOps process belongs to the same category of the agile process [56], given that DevOps concept originated in agile software development [10] [56].

The reviewed studies rarely discuss explicitly a DevOps process, which creates a challenge affecting the research in this field, not having a standard definition and being subject to publications bias.

### 4.13.3 RQ3: What are the DevOps related practices?

Section 4.11 presents set DevOps practices extracted from studies reporting explicitly the subject. According to [78] DevOps is a set of practices that is trying to bridge developer-operations which imply more attention to related practices as is possible to observe in Appendix A where is highlighted the continuous practices: continuous delivery, continuous integration, continuous deployment, continuous monitoring, and continuous testing.

Therefore, we found it interesting to relate the practices with the identified DevOps process phases. For this purpose, we utilize the most cited practices and identified process phases and relate them, following the proposal of [12]. Although some adaptations were necessary to fit with our idea, table 4.5 and figure 4.6.

| | Process | Practices |
|---|---|---|
| **Software Development and Delivery** | Plan | Continuous Integration |
| | Code | |
| | Build | |
| | | |
| | Build | Continuous Testing |
| | Test | |
| | Code | Continuous Delivery |
| | Release | |
| **Software Monitor and Maintenance** | Deploy | Continuous Deployment |
| | | |
| | Deploy | Continuous Monitoring |
| | Operate | |
| | Monitor | |

**Figure 4.5:** DevOps Components

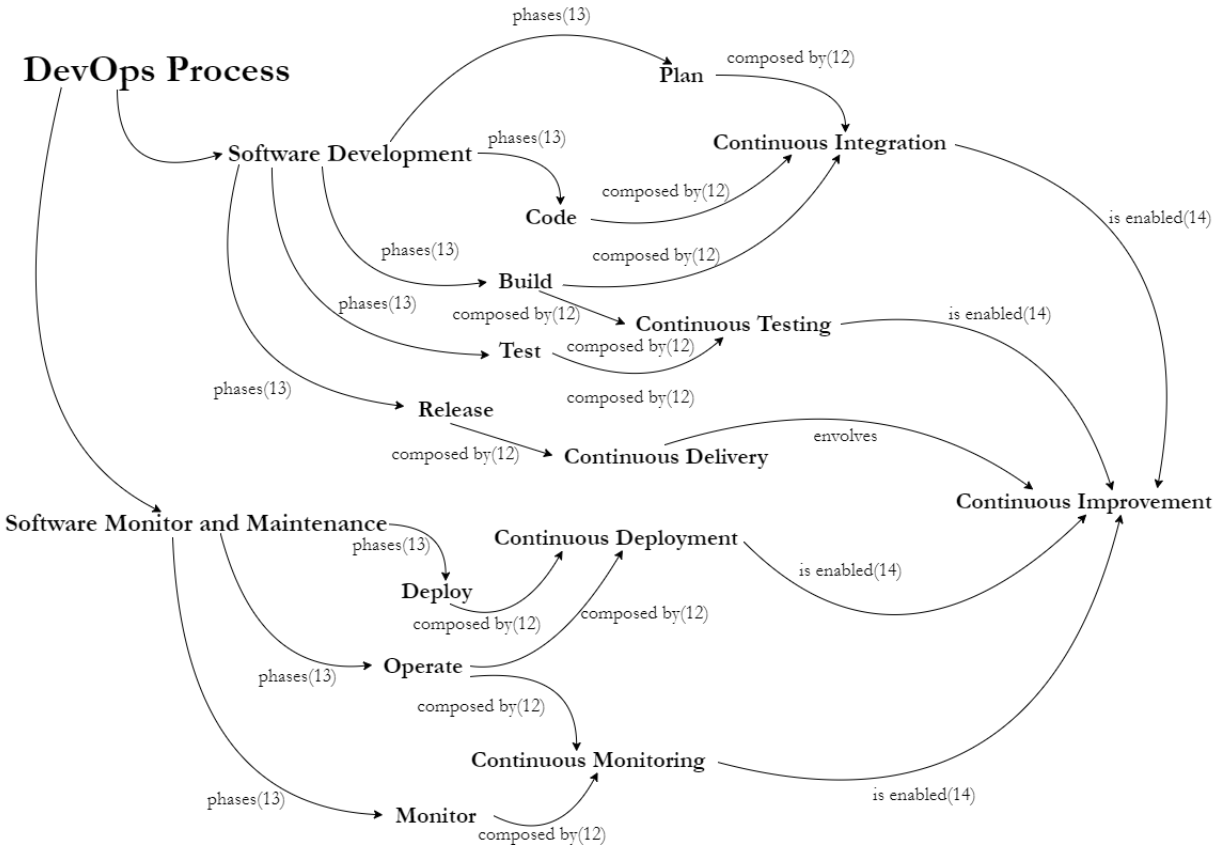


**Figure 4.6:** DevOps components conceptual model

Furthermore, we reuse the knowledge obtained by studying the DevOps process and how it is related to Agile software development. [59] stated that both DevOps and agile have an emphasis on continuous integration, testing, and delivery of working software, however, DevOps adds to agile software development the emphasis on automating the analysis of applications, as well as ensure the continuous communications between the development and operations teams.

### 4.13.4 RQ4: What are the key roles for DevOps?

The SLR reported several roles (Table 7) in which emerge the role of DevOps engineer Gill et al., [57] establishing the link between the development and operations. This implied the necessity of organizations to define a new role of DevOps engineer to provide the new need for crucial technical support.

Based on the identified practices we found interesting to relate some of them with DevOps roles using a *RACI matrix model* [2]. Table 4.7.

| ROLES / DevOps Practices | DevOps Engineer | Developer | Tester | QA Engineer | Application Arquitect | Project Manager | Release Manager | Network Administrator | Database Administrator | Infrastruture Management | System Administrator | System Engineer |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Development | | | | | | | Operation | | | | |
| Continuous Delivery | A | R | | | C | I | | | | | | |
| Continuous Deployment | | | | | | | | C | S | A | R | I |
| Continuous Integration | C | R | | | A | | | | | | | |
| Continuous Monitoring | | | | | | | | A | C | R | I | |
| Continuous Testing | | I | A | R | S | | C | | | | | |

| | | |
|---|---|---|
| D | Driver | Assists those who are responsible for a task. |
| R | Responsible | Assigned to complete the task or deliverable. |
| A | Accountable | Has final decision-making authority and accountability for completion. Only 1 per task. |
| S | Support | Provides support during implementation. |
| C | Consulted | An adviser, stakeholder, or subject matter expert who is consulted before a decision or action. |
| I | Informed | Must be informed after a decision or action. |

**Figure 4.7:** DevOps practices and roles

[2] RACI Matrix: https://www.vertex42.com/ExcelTemplates/raci-matrix.html (Accessed 04/04/20).

# 5

# DevOps Maturity Model Proposal

## Contents

As described in section Research Methodology and according to DSRM develop and design phase, This section shows a comprehensive overview of the steps to develop the MM as well as the proposed model.

In the existing literature, it was quite often chosen to come up with their own definitions for the DevOps maturity levels. Although CMMI is widely known and used as a standard to define the levels of maturity, we propose to follow a different approach in which levels are defined by the well known as ISO/IEC 3300x.

## 5.1 Objective

The main goal of this proposed solution is to develop a maturity model to assess the current state of DevOps adoption in the organization and provide instruction to achieve a higher level of DevOps maturity.

In order to achieve this objective two tasks must be completed:

- The creation of a Process Reference Model (PRM)
- The creation of a Process Assessment Model (PAM)

By integrating these two models, our maturity model should be able to become a useful artefact for organizations to use as a guide to assess the maturity level of DevOps implementation. This would enable them to deliver extensive internal benefits, for instance in build automation, more collaboration between teams, automation of production cycle, and aligning the customer requirements with the product.

In order to substantiate the validity of the artefacts produced, they will have the support of standards, procedures and methods accepted by the community as questionnaires or interviews answered by experts whether they are researchers or practitioners in the field of DevOps.

## 5.2 PRM for DevOps

The previous sections provide the necessary material to create a first version of the DevOps maturity model. In this case, it was chosen to use ISO/IEC framework as a starting point, based on the arguments given in section 5.1. This framework will be combined with the processes found in the literature. The tables below show the result of PRM applied to DevOps processes identified according to the SLR. It should be noted that the processes are divided into two major groups, namely software development and delivery which comprises the phases: Plan, Code, Build, Test, and Release. Next, software operation and monitor which is composed of Deploy, Operate, and Monitor.

**Table 5.1:** PRM for Process Plan

| **Process ID** | SDD.01 |
|---|---|
| **Title** | Plan |
| **Context** | This step is usually dealt with by project managers in collaboration with the team and exploiting project management tools. |
| **Purpose** | Activity planning and task scheduling for the current release. |
| **Outcomes** | As result of successful implementation of this process:<br>1. Product requirement collected and documented.<br>2. Features are planned.<br>3. Project milestones are defined.<br>4. User experience designed.<br>5. Scope defined.<br>6. Stakeholders are identified. |

**Table 5.2:** PRM for Process Code

| **Process ID** | SDD.02 |
|---|---|
| **Title** | Code |
| **Context** | Code development and code review. |
| **Purpose** | Developers are the most closely involved in this activity using integrated development environment (IDE) and tools for source code management. |
| **Outcomes** | As result of successful implementation of this process:<br>1. Code is developed.<br>2. Code is reviewed.<br>3. Code is tested.<br>4. Source code is organized.<br>5. Features are documented. |

**Table 5.3:** PRM for Process Build

| **Process ID** | SDD.03 |
|---|---|
| **Title** | Build |
| **Context** | When source code is converted into a stand-alone form that can be run on a computer system. In this activity are involved various professional figures. |
| **Purpose** | The process of creating an executable artefact from input such as source code and configurations. |
| **Outcomes** | As result of successful implementation of this process:<br>1. Source code is compiled, and all files required for execution are packaged.<br>2. Build and integrations tests are performed.<br>3. Version control repository is created.<br>4. Executable artefact is created. |

**Table 5.6:** PRM for Process Deploy

| **Process ID** | SOM.01 |
|---|---|
| **Title** | Deploy |
| **Context** | Documentation about releases (for example new features introduced), new final version of executables files or packages, logs and metrics from release orchestration tools. |
| **Purpose** | The deployment process refers to releasing that code to customers<br>Deals with installation and execution of the new software release in the production environment and infrastructure. |
| **Outcomes** | As result of successful implementation of this process:<br>1. Application code is deployed on all production servers.<br>2. New features are documented.<br>3. Automation management, maintenance of configurations.<br>4. Artefact repository.<br>5. Source code repository.<br>6. Configuration management data architecture. |

**Table 5.4:** PRM for Process Test

| **Process ID** | SDD.04 |
|---|---|
| **Title** | Test |
| **Context** | Building the test harness and/or scaling up test execution in larger build systems. |
| **Purpose** | When source code is converted into a stand-alone form that can be run on a computer system. In this activity are involved various professional figures. |
| **Outcomes** | As result of successful implementation of this process:<br>1. Function and non-functional acceptance tests are performed.<br>2. Unit test is performed.<br>3. Component test is performed.<br>4. Deployment tests are performed.<br>5. User acceptance tests are performed.<br>6. Automated acceptance tests are performed. |

**Table 5.5:** PRM for Process Release

| **Process ID** | SDD.05 |
|---|---|
| **Title** | Release |
| **Context** | Documentation about releases (for example new features introduced), new final version executable files or packages, logs and metrics from release orchestration tools. |
| **Purpose** | Triggered when a new version of software is ready to be released to end users.<br>Setting up and maintaining pipeline for deployment and release of a new version of a software project.<br>Delivers the system to users, either as packaged software or deploying it into a production or staging environment (staging environment is a testing enviroment identical to the production environment). |
| **Outcomes** | As result of successful implementation of this process:<br>1. Created and maintained environment configurations and controls.<br>2. Release and deployment plans are defined with customers/stakeholders.<br>3. Release roadmap and cycle time is determined.<br>4. Code integrity is verified.<br>5. Code conflicts are solved. |

**Table 5.7:** PRM for Process Operate

| **Process ID** | SOM.02 |
|---|---|
| **Title** | Operate |
| **Context** | The activity that maintains and adapts the infrastructure in which the software is running. |
| **Purpose** | Data generated by the software, logs from the tools involved in this stage and system logs from (physical/virtual) servers. |
| **Outcomes** | As result of successful implementation of this process:<br>1. Capacity and resources are planned and managed.<br>2. Infrastructure is installed.<br>3. Configuration and management of infrastructures changes.<br>4. Security checks are performed. |

**Table 5.8:** PRM for Process Monitor

| | |
|---|---|
| **Process ID** | SOM.03 |
| **Title** | Monitor |
| **Context** | Phase takes place parallel to the Operate phase and consists of data collection, analysis, and feedback to the start of the pipeline and to other phases as needed. |
| **Purpose** | This activity the software in production is monitored by mainly sysadmin, operators and other managing the project. |
| **Outcomes** | As result of successful implementation of this process:<br>1. Applications performance are monitored.<br>2. Services performance are monitored.<br>3. Application log is monitored.<br>4. Code integrity is verified.<br>5. User activity monitored for improper behaviour of the system.<br>6. Information about issues from a software release in production is identified and collected.<br>7. Data collected and analysed, and feedback is provided. |

The process reference described above in terms of its purpose and associated outcomes can help companies get a better understanding of their DevOps processes.

Despite our reliance on the usefulness and applicability of this set of reference processes for DevOps, these should not be seen as the only and exclusive processes in terms of outcomes, being that they can be improved and adapted for achieving best results in DevOps adoption.

## 5.3 PAM for DevOps

PRM are always related to PAM which holds all indicators to determine the maturity of the processes of the reference model. In this section, we will cover the processes identified and detailed regarding the process assessment model adopted and what are the conditions to achieve a certain maturity level for each process.

The proposal is to assess the DevOps processes in organizations is based on ISO/IEC 330xx. We chose this family standard because it is a global reference and provides requirements for the construction and verification of process reference models, process assessment models, and maturity models. It is an international standard and because of its greater adaptability for the purpose of determining the current capability of organizations' processes, as well as establishing priorities for process improvement. Additionally, a design science research project conducted by [87] advocates that "the external validity of an artefact can be improved with the use of International Standards" and concludes that the project in question "demonstrated the compelling role of International Standards to check research relevance during artefact design, development and evaluation". Our process assessment model follows the example of ISO/IEC 33072, the process capability assessment model for information security management, and is structured in accordance with the requirements of ISO/IEC 33004, also used by ISO/IEC 33072.

Close to the point of building a DevOps maturity model, it is necessary to identify relevant processes in this context. This has been done before using a systematic literature review.

The study resulted in the following processes that are grouped into two major action fields Figure 5.1

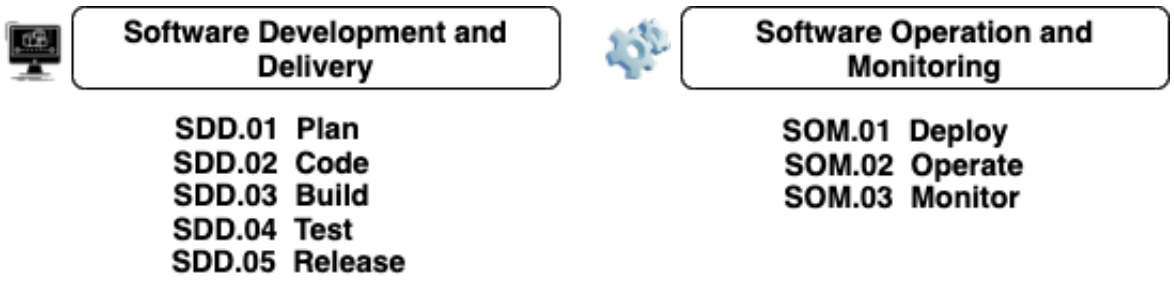


**Figure 5.1:** Process Reference Model (PRM) including DevOps processes identified from our previous research effort and grouped by action fields. Adapted from [3]

It is important to emphasize that PAM is a two-dimensional model concerning the process dimension and capability dimension. The representation of what constitutes the process dimension of the PAM for eight processes are presented in the following tables.

**Table 5.9:** SDD.01 Plan

| | |
|---|---|
| **Process ID** | SDD.01 |
| **Title** | Plan |
| **Context** | This step is usually dealt with by project managers in collaboration with the team and exploiting project management tools. |
| **Purpose** | Activity planning and task scheduling for the current release. |
| **Outcomes** | As result of successful implementation of this process:<br>1. Product requirement collected and documented.<br>2. Stakeholders are identified.<br>3. Project milestones are defined.<br>4. User experience is designed.<br>5. Scope defined.<br>6. Features are planned. |
| **Base Practices** | SDD.01.1 – Identify and collect product requirement [Outcome 1]<br>SDD.01.2 – Identify Stakeholders [Outcome 2]<br>SDD.01.3 – Set project milestones [Outcome 3]<br>SDD.01.4 – Design user experience [Outcome 4]<br>SDD.01.5 – Set scope[Outcome 5]<br>SDD.01.6 – Plan features [Outcome 6] |
| **Inputs** | |
| Document product requirements [Outcome 1]<br>Envisioning software [Outcome 4]<br>Task management [Outcome 3]<br>Schedules [Outcome 2] | |
| **Outputs** | |
| Product requirements specification document [Outcome 1]<br>Vision of the project [Outcome 4] | |

**Table 5.10:** SDD.02 Code

| **Process ID** | SDD.02 |
|---|---|
| **Title** | Code |
| **Context** | Code development and code review. |
| **Purpose** | Developers are the most closely involved in this activity using integrated development environment (IDE) and tools for source code management. |
| **Outcomes** | As result of successful implementation of this process:<br>1. Code is developed.<br>2. Code is reviewed.<br>3. Code is tested.<br>4. Source code is organized.<br>5. Features are documented. |
| **Base Practices** | SDD.02.1 – Code development [Outcome 1]<br>SDD.02.2 – Code review[Outcome 2]<br>SDD.02.3 – Write and test the code [Outcome 3]<br>SDD.02.4 – Manage source code[Outcome 4]<br>SDD.02.4 – Document features [Outcome 4] |
| **Inputs** | |
| Organize source code [Outcome 4]<br>Write tests [Outcome 3]<br>Peer review code [Outcome 2] | |
| **Outputs** | |
| Feature Documents [Outcome 2]<br>Executable artefact [Outcome 4] | |

**Table 5.11:** SDD.03 Build

| **Process ID** | SDD.03 |
|---|---|
| **Title** | Build |
| **Context** | When source code is converted into a stand-alone form that can be run on a computer system. In this activity are involved various professional figures. |
| **Purpose** | The process of creating an executable artefact from input such as source code and configurations |
| **Outcomes** | As result of successful implementation of this process:<br>1. Source code is compiled, and all files required for execution are packaged.<br>2. Build and integration tests are performed.<br>3. Version control repository is created.<br>4. Executable artefact is created. |
| **Base Practices** | SDD.03.1 – Compile source code and package all execution files [Outcome 1]<br>SDD.03.2 – Perform build and integration tests [Outcome 2]<br>SDD.03.3 – Create version control repository [Outcome 3]<br>SDD.03.4 – Create executable artefact[Outcome 4] |
| **Inputs** | |
| Continuous integration [Outcome 2]<br>Version control [Outcome 3]<br>Artefact creation [Outcome 4] | |
| **Outputs** | |
| Build status [Outcome 2]<br>Executable artefact [Outcome 4]<br>Quality assurance [Outcome 5,6] | |

**Table 5.12:** SDD.04 Test

| | |
|---|---|
| **Process ID** | SDD.04 |
| **Title** | Test |
| **Context** | Building the test harness and/or scaling up test execution in larger build systems. |
| **Purpose** | When source code is converted into a stand-alone form that can be run on a computer system. In this activity are involved various professional figures. |
| **Outcomes** | As result of successful implementation of this process:<br>1. Functional and non-functional acceptance tests are performed<br>2. Unit test is performed<br>3. Component test is performed<br>4. Deployment tests are performed<br>5. User acceptance tests are performed<br>6. Automated user acceptance tests are performed |
| **Base Practices** | SDD.04.1 – Run functional and no-functional acceptance tests [Outcome 1]<br>SDD.04.2 – Develop unit tests [Outcome 2]<br>SDD.04.3 – Perform component tests [Outcome 3]<br>SDD.04.4 – Perform deployment tests[Outcome 4]<br>SDD.04.5 – Perform user acceptance tests [Outcome 5]<br>SDD.04.6 – Perform automated user acceptance tests [Outcome 6] |
| **Inputs** | |
| Timely feedback [Outcome 1]<br>User acceptance tests [Outcome 5,6]<br>Staging test[Outcome 4]<br>Performance test [Outcome 5]<br>Develop test cases[Outcome 5]<br>Write automated tests [Outcome 2,5] | |
| **Outputs** | |
| Quality checks [Outcome 1,5]<br>Performance measurement [Outcome 3]<br>Quality assurance [Outcome 5,6] | |

**Table 5.13:** SDD.05 Release

| | |
|---|---|
| **Process ID** | SDD.05 |
| **Title** | Release |
| **Context** | documentation about releases (for example new features introduced), new final version of executable files or packages, logs and metrics from release orchestration tools |
| **Purpose** | Triggered when a new version of software is ready to be released to end users. Setting up and maintaining the pipeline for deployment and release of a new version of software. |
| **Outcomes** | As result of successful implementation of this process:<br>1. Created and maintained environment configuration and controls<br>2. Release and deployment plans are defined with customers/stakeholders<br>3. Release roadmap and cycle time is determined<br>4. Code integrity is verified<br>5. Code conflicts are solved |
| **Base Practices** | SDD.05.1 – Create and maintain environment configuration and controls [Outcome 1]<br>SDD.05.2 – Define release and deployment plans with customers/stakeholders [Outcome 2]<br>SDD.05.3 – Determine the release roadmap and cycle time [Outcome 3]<br>SDD.05.4 – Verify code integrity [Outcome 4]<br>SDD.05.4 – Solve code conflicts [Outcome 5] |
| **Inputs** | |
| Release planning [Outcome 1]<br>Test and verify release [Outcome 2]<br>Delivery the release [Outcome 4]<br>Release management [Outcome 5]<br>Release approval[Outcome 5]<br>Release automation[Outcome 5] | |
| **Outputs** | |
| Track changes [Outcome 2]<br>System availability [Outcome 3] | |

**Table 5.14:** SMM.01 Deploy

| **Process ID** | SOM.01 |
|---|---|
| **Title** | Deploy |
| **Context** | Documentation about releases (for example new features introduced), new final version of executable files or packages, logs and metrics from release orchestration tools. |
| **Purpose** | The deployment process refers to releasing that code to customers. Deals with installation and execution of the new software in the production environment and infrastructure. |
| **Outcomes** | As a result of successful implementation of this process:<br>1. Application code is deployed on all production servers.<br>2. New features are documented.<br>3. Automation management, maintenance of configurations.<br>4. Artefact repository.<br>5. Source code repository.<br>6. Configuration management data architecture. |
| **Base Practices** | SOM.01.1 – Deploy application code on all production servers [Outcome 1]<br>SOM.01.2 – Document new features [Outcome 2]<br>SOM.01.3 – Automate management, and maintenance of configurations [Outcome 3]<br>SOM.03.4 – Artefact repository [Outcome 4]<br>SOM.01.4 – Source code repository [Outcome 5]<br>SOM.01.5 – Configure and manage data architecture [Outcome 6] |
| **Inputs** | |
| Deployment of application code on all servers [Outcome 1]<br>Containerization [Outcome 2]<br>Infrastructure configuration information [Outcome 4]<br>Data recovery [Outcome 5] | |
| **Outputs** | |
| Track changes [Outcome 2]<br>System availability [Outcome 3] | |

**Table 5.15:** SOM.02 Operate

| | |
|---|---|
| **Process ID** | SOM.02 |
| **Title** | Operate |
| **Context** | documentation about releases (for example new features introduced), new final version of executable files or packages, logs and metrics from release orchestration tools |
| **Purpose** | Triggered when a new version of software is ready to be released to end users. Setting up and maintaining the pipeline for deployment and release of a new version of a software project. |
| **Outcomes** | As a result of successful implementation of this process:<br>1. Capacity and resources are planed and managed<br>2. Infrastructure is installed<br>3. Configuration and management of infrastructures changes<br>4. Security checks are performed |
| **Base Practices** | SOM.02.1 – Plan and manage capacity and resources [Outcome 1]<br>SMM.02.2 – Infrastructure installation [Outcome 2]<br>SOM.02.3 – Perform security checks [Outcome 3]<br>SOM.02.4 – Verify the code integrity [Outcome 4] |
| **Inputs** | |
| Capacity planning [Outcome 1]<br>Infrastructure installation and procedure and changes [Outcome 2]<br>Service deployment [Outcome 4]<br>Data recovery [Outcome 5]<br>Log/backup management [Outcome 6]<br>Database management [Outcome 4] | |
| **Outputs** | |
| Track changes [Outcome 2]<br>System availability [Outcome 3]<br>System security [Outcome 6] | |

**Table 5.16:** SOM.03 Monitor

| **Process ID** | SOM.03 |
|---|---|
| **Title** | Monitor |
| **Context** | Phase takes place parallel to the Operate phase and consists of data collection, analysis, and feedback to the start of the pipeline and to other phases as needed. |
| **Purpose** | This activity the software in production is monitored by mainly sysadmin, operators and other managing the project. |
| **Outcomes** | As result of successful implementation of this process:<br>1. Applications performance are monitored.<br>2. Services performance are monitored.<br>3. Application log is monitored.<br>4. Code integrity is verified.<br>5. User activity monitored for improper behaviour of the system.<br>6. Information about issues from a software release in production is identified and collected.<br>7. Data collected and analysed, and feedback is provided. |
| **Base Practices** | SOM.03.1 – Monitor applications performance. [Outcome 1]<br>SOM.03.2 – Monitor services performance. [Outcome 2]<br>SOM.03.3 – Monitor application log. [Outcome 3]<br>SOM.03.4 – Verify code integrity [Outcome 4]<br>SOM.03.5 – Monitor user activity for improper behaviour of the system. [Outcome 5]<br>SOM.03.6 – Identify and collect information about issues from a software release in production. [Outcome 6]<br>SOM.03.7 – Collect and analyse data and provide feedback. [Outcome 7] |
| **Inputs** | |
| Applications performance monitoring [Outcome 1]<br>Services performance monitoring [Outcome 2]<br>Log monitoring [Outcome 3]<br>End-user experience [Outcome 5]<br>Incident management [Outcome 6]Provide feedback [Outcome 7] | |
| **Outputs** | |
| Performance measurement reports [Outcome 1,2]<br>System availability status [Outcome 3]<br>System security status [Outcome 6]<br>Systems status report [Outcome 3] | |

## 5.4 Maturity Model

Sections 5.2 and 5.3 provided necessary material to create a first version of the DevOps maturity model. In this case, it was chosen to use the ISO/IEC 330xx family standard. Therefore, the implementation of the model will help improve operational efficiency and increase the visibility of the processes. The strength of this model is that using tools of ISO/IEC helped on the identification and create the reference of the DevOps processes that were used as the base for the following phases.

This framework combines PRM and PAM as the foundations. The Maturity Model is composed of six maturity levels against 8 dimensions of DevOps processes: **'Plan'**, **'Code'**, **'Build'**, **'Test'**, **'Release'**, **'Deploy'**, **'Operate'** and **'Monitor'**.

Within PRM and PAM, it is possible to see the full structure and description of the dimensions for the specific maturity. further, our proposed model includes six levels, which the first or initial identified by 0 that determines whether the assessment continues or not.

### 5.4.1 Process Performance Indicators

There are two types of process performance indicators: Base Practice (BP) indicators and Input/Output (IO) indicators. Process performance indicators relate to individual processes defined in the process dimension of the PAM and are chosen to explicitly address the achievement of the defined process outcomes. Evidence of performance of the base practices, and the presence of inputs/outputs with their expected characteristics, provide objective evidence of the achievement of the process outcomes. A base practice is an activity that addresses the purpose of a particular process. Consistently performing the base practices associated with a process will help the consistent achievement of its purpose. A coherent set of base practices is associated with each process in the process dimension. The base practices are described at an abstract level, identifying "what" should be done without specifying "how". Implementing the base practices of a process should achieve the basic outcomes that reflect the process purpose. Base practices represent only the first step in building process maturity, but the base practices represent the unique, functional activities of the process, even if that performance is not systematic.

In this particular PAM the base practices have been used as a vehicle to link the outcomes of each process in the PRM with the requirements defined for that process in ISO/IEC 27001. This has been achieved using the following strategy:

- Singular requirements from ISO/IEC 27001 have been identified and assigned a unique identifier (process number plus sequential numbering within the sub-clause).
- Each process outcome has been linked to a single base practice.

This approach provides insight on how the singular requirements from ISO/IEC 27001 contribute to the

achievement of the process purpose and outcomes. The performance of a process requires inputs and produces outputs that are identifiable and usable in achieving the purpose of the process. In this assessment model, each input/output has a defined set of example characteristics that may be used when reviewing the input/output to assess the effective performance of a process. Input/output characteristics may be used to identify the corresponding input/output produced/used by the assessed organization [49].

### 5.4.2 Maturity Levels

The aim of maturity levels is to classify organisations according to their ability to control their various processes. Therefore, they are defined on a six point ordinal scale that enables maturity to be assessed from the bottom of the scale, Incomplete, through to the top end of the scale, Innovating. The scale represents increasing maturity of the implemented process, from failing to achieve the process purpose through to continually improving and able to respond to organizational change.

- **Level 0: Incomplete process**

  The process is not implemented or fails to achieve its process purpose. At this level, there is little or no evidence of any systematic achievement of the process purpose.

- **Level 1: Performed process**

  The implemented process achieves its process purpose.

- **Level 2: Managed process**

  The previously described Performed process is now implemented in a managed fashion (planned, monitored and adjusted) and its work products are appropriately established, controlled and maintained.

- **Level 3: Established process**

  The previously described Managed process is now implemented using a defined process that is capable of achieving its process outcomes.

- **Level 4: Predictable process**

  The previously described Established process now operates predictively within defined limits to achieve its process outcomes. Quantitative management needs are identified, measurement data are collected and analysed to identify assignable causes of variation. Corrective action is taken to address assignable causes of variation.

- **Level 5: Innovating process**

  The previously described Predictable process is now continually improved to respond to change aligned with organizational goals.

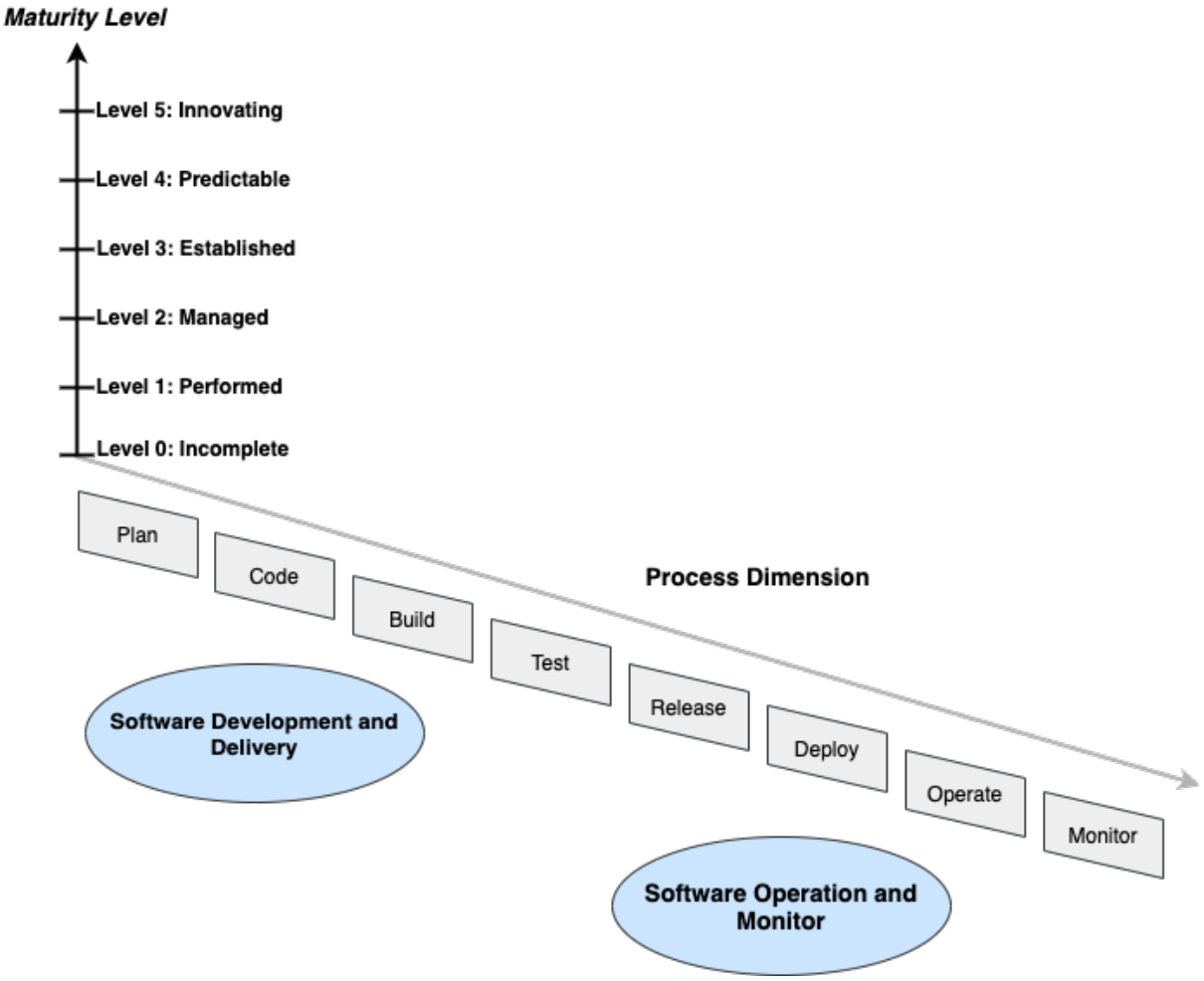


**Figure 5.2:** Maturity Model

# 6

# Demonstration

## Contents

The section covers the description of the demonstration stage of DSRM. The approach put forward should be called into question in order to prove its efficiency in solving the problem specified in the research problem section.

The objective of the DevOps maturity model is to be applicable in real scenarios, thus it is important to interview experts and practitioners in the DevOps field in which processes outcomes they think are necessary and add value for successful implementation of DevOps in the organization.

Regarding the assessment, we met with Company A's DevOps team in order to perform the assessment following PAM addressing all the processes identified. Although PAM is carried out taking into account process best practices, inputs, and outputs, the assessment only took into account the outcomes as it was the first interaction with the processes.

The interview taken with the expert followed a questionnaire, but we could divert if something came up that had to be explored further. The interview was done in one round comprised of questions about DevOps and processes purpose and outcomes associated with different stages of DevOps processes. The data acquired would be very important further to enrich the existent model, although it was not possible to perform more rounds of interviews.

| **Maturity Level**<br>**Description** | **Questionnaire**<br>**Statements** |
|---|---|
| **0 Incomplete process**<br>**The process is not implemented or fails to achieve its process purpose.** | - At this level, there is little or no evidence of any systematic achievement of the process purpose. |
| **1 Performed process**<br>**The implemented process achieves its process purpose** | - Functional and non-functional tests are performed.<br>- Unit test is performed.<br>- Component tests are performed<br>- Deployment tests are performed<br>- User acceptance tests are performed<br>- Automated acceptance tests are performed |

**Table 6.1:** Questionnaire Construction for SDD.04 Test (maturity levels 0 and part of 1)

To arrange the questionnaire, the maturity level descriptions of the DevOps Maturity Model were validated following ISO/IEC standard. The examination resulted in the realization that a rating value could be computed for the maturity level by collecting and then combining the percentages value for each outcome. Table 6.1 displays an example of how the questionnaire statements were derived for the maturity model of process SDD.04 Test.

The demonstration counted on the assessment of plan, code, build, test, release, deploy, operate, and monitor, performed to conclude what capability level the company was. At this stage of the study, superior levels were considered out of the scope of this assessment. To determine whether the processes were implemented or not, the classification used by the ISO/IEC 330XX family of standards was adopted. That is, with the Company A DevOps team self-assessment within the respective standard scale (Not Achieved, Partially Achieved, Largely Achieved and Fully Achieved) to evaluate process purpose and outcomes. It works as follow:

- If the Process Purpose rating is lower than Largely Achieved, i.e. Not Achieved or Partially Achieved, the capability level is immediately considered 0 (however the rating continues for outcomes).
- For outcomes, each is independently evaluated. After all, are evaluated, a median of the results is made to arrive at a representative value of the outcomes.
- Finally, we compare the process purpose classification with the outcomes classification where the inferior classification is the one that persists. If the final rating is Largely Achieved or Fully Achieved it means that capability level 1 has been reached. Otherwise, the capability level is 0.

**Table 6.2:** Rating Scale

| Rating Scale | Corresponding percentages |
|---|---|
| **Not achieved (N)** | 0 to $\leq$ 15% |
| **Partially achieved** | $\geq$ 15% to $\leq$ 50% |
| **Largelly achieved** | $\geq$ 50% to $\leq$ 85% |
| **Fully achieved** | $\geq$ 85% to $\leq$ 100% |

## 6.1 Results

The meeting took about one hour and fifteen minutes, it started with a brief presentation of all participants gathering some general data, like the background and years of experience of the interviewee. Then a presentation of the framework was made known, detailing high-level steps taken to achieve it.

The next step following up the meeting was framework assessment with two members of DevOps team of the Company A, and it was done assessing each of processes purposes and outcomes and evaluating them accordingly to the process attribute rating scale defined by ISO/IEC 330xx family, which is known as ”judgment of the degree of achievement of the process attribute for the assessed process”. Some of the results obtained are described in figures 6.1 and tables 6.1, and 6.2. Therefore, the results are described according to the structure that was used in the interview, although it will not be presented in the same order as the interview, starting with Plan and Monitor, followed by Test and Operate.

**Table 6.3:** Assessment results

| | | Level | | | |
|---|---|---|---|---|---|
| | **Outcomes** | **Not Achieved** | **Partially achieved** | **Largely achieved** | **Fully achieved** |
| Plan | **1. Product requirement collected and documented** | | | | X |
| | **2. Stakeholders are identified** | | | X | |
| | **3. Project milestones are defined** | | | X | |
| | **4. User experience is designed** | | | | X |
| | **5. Scope defined** | | | X | |
| | **6. Features are planned** | | | | X |
| Monitor | **1. Applications performance are monitored** | | | X | |
| | **2. Services performance are monitored** | | | X | |
| | **3. Application log is monitored** | | | X | |
| | **4. Code integrity is verified** | | | | X |
| | **5. User activity monitored for improper behaviour of the system** | | | | X |
| | **6. Information about issues from a software release in production is identified and collected** | | | | X |
| | **7. Data collected and analysed, and feedback is provided** | | | X | |

The first process that was assessed in the interview was the Plan that is composed of six outcomes, where we can observe the highest level obtained was F and the lower was L. According to the interviewee, the lowest levels are related to the non-fulfilment of some activities that involve the outcome by all teams involved in the process. The interviewee explained this as;

*"Not all features are planned at this stage and not all project milestones are defined"*

The second process regarding DevOps is the Operate, which belongs to the major field of software operation and monitoring. it is composed by seven outcomes of which four obtained L as rating and the other three obtained F. Although it could be considered as a good, improvements have to be made to allow the fulfilment of all process outcomes. as the interviewee said:

*"Although we have this outcome implemented, we are not mature enough to assure the complete fulfilment of them. we still have to improve it"*

Following the same approach applied for the two first processes, the assessment was made for the processes Test and Operate retrieved similar results, but the outcome 2 from Test stand out for the lower rating obtained,

**Table 6.4:** Assessment

| | | Level | | | |
|---|---|---|---|---|---|
| | **Outcomes** | **Not Achieved** | **Partially achieved** | **Largely achieved** | **Fully achieved** |
| Test | **1. Functional and non-functional tests are performed** | | | | X |
| | **2. Unit test is performed** | | X | | |
| | **3. Component test is performed** | | | X | |
| | **4. Deployment tests are performed** | | | | X |
| | **5. User acceptance tests are performed** | | | X | |
| | **6. Automated acceptance tests are performed** | | | X | |
| Operate | **1. Capacity and resources are planned and managed** | | | | X |
| | **2. Infrastructure is installed** | | | | X |
| | **3. Configuration and management of infrastructures changes** | | | | X |
| | **4. Security checks are performed** | | | | X |

as it can be observed in table 6.2. According to the interviewee, this result is related to the lack of alignment between teams and the accomplishment of some tasks related to this outcome.

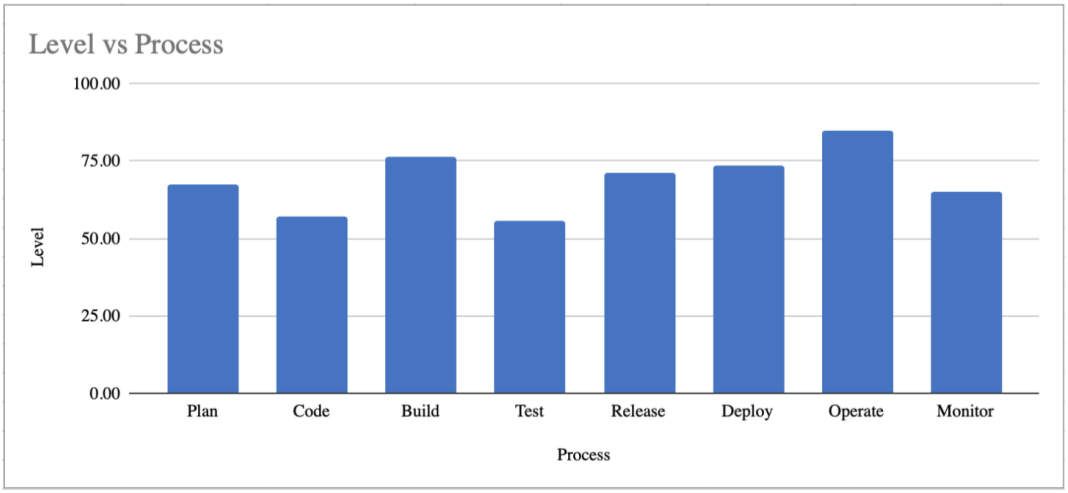


**Figure 6.1:** First demonstration

The result provided interesting information on their own basis. However, it is important to look to results to find what conclusions can be taken from them. Therefore, The first aspect that becomes clear when looking at the results from the figure 6.1 gives an overview of the average levels, which stand out operate in one hand as the process with the maximum percentage of implementation and in the opposite side, test and code as the

ones with the lower percentage of implementation. This indicates that although the Company has implemented all processes, they are still improving the quality of deliverance of them.

# 7

# Evaluation and Communication

## Contents

The section covers the description of the evaluation and communication stages of DSRM.

Following the Pries-Heje et al approach, which presents the importance of an ex-ante perspective, with evaluation occurring both prior to the construction of an artefact IS, and ex-post evaluation, that is, the evaluation that takes place after the artefact has been built [79]. For DSRM, Venable distinguishes two main forms [80]:

- Artificial evaluation evaluates a solution technology in a contrived, non-real way.
- Naturalistic evaluation enables a researcher to explore how well or poorly a solution technology works in its real environment – the organization.

In our study, an artificial evaluation was performed by applying the ISO/IEC 330xx family of standards to prove that it is possible to build a maturity model specifically for DevOps, using an approach based on design science research. Feasibility was demonstrated for the processes Plan, code, Build, Test, Release, Deploy, Operate, and Monitor, suggesting that it is feasible to follow the same steps for all other processes included in our PRM. A second artificial evaluation was conducted by checking the applicability of our framework into an excel questionnaire.

In a nutshell, the results prove that it is possible to build a maturity model for DevOps grounded in a well-known standards and methodologies. Finally, it can be used by organizations to help them evaluate their maturity in adopting DevOps.

## 7.1 Communication

In harmony with the DSRM's communication proposal, we aim to communicate our artefacts to the applicable audience. For this purpose, we pursued two different ways to deliver the communication proposal described as follow:

- Publish papers through scientific journals or conferences.
- Communicate the work-study through the dissertation itself.

To reach a broader communication in our work, a paper entitled "A Systematic Literature Review about Processes and Roles in DevOps", consists of the literature review for DevOps concepts, processes, practices, and roles. It brought very important contributions in the identification of the processes that guided the PRM and consequently PAM, as well as explore the interesting relations between DevOps processes, practices and roles. The was submitted to the International Journal of Agile Systems and Management (IJASM). The paper still awaits confirmation of acceptance.

Finally, the final dissertation report, containing all the content related to DevOps maturity model, will be presented, discussed, and evaluated by a qualified jury to ensure its reliability and the quality of the scientific contribution. Subsequently, the work will be shared with the public.

# 8

# Conclusion

## Contents

## 8.1 Contributions

To conduct this research, we followed DSRM process, that comprises 6 phases of development. First, we identified the problem, the lack of DevOps maturity models to guide companies to assess the maturity of their DevOps adoption. The main objectives propose a DevOps maturity model to assess the current state of DevOps adoption in an organization and provide guidance to achieve higher levels of DevOps maturity. To address this problem, a literature review was conducted addressing DevOps concepts, processes, practices, and roles and as result of this, we found 32 concepts, 9 processes, 33 practices, and 20 roles as it can be observed in tables 4.4, 4.6, 4.7, and appendix A. From this stands out the concepts of automation, knowledge sharing, continuous delivery, infrastructure as a code, communication and collaboration as the most frequently cited ones; Plan, Test, Release, Deploy, and Monitor for the processes; Continuous Delivery, Continuous Integration, Continuous Deployment, Continuous Monitoring, and Continuous Testing for practices; and finally, as the most frequently cited roles, DevOps Engineer, Developer, IT Manager, Network Administrator, and Cloud Engineer.

The SLR also allowed us to explore and bring to this research our contribution about the relation between DevOps processes and some practices, as shown in figures 4.5, and 4.6, as well as the relationship between practices and roles (figure 4.7). Thus, the identified DevOps processes were the foundation for PRM and consequently served as the base for PAM.

Once the processes were identified through SLR, thus constituting our PRM, we moved to PAM where we had the processes Plan, Code, Build, Test, Release, Deploy, Operate, and Monitor with their respective details following the procedure and structure of ISO/IEC 330xx family standard.

The study aimed to create a suitable maturity model to assess and improve DevOps environment in an organization. Therefore, one major question was raised.

***RQ1. What is a suitable DevOps Maturity framework for assessing the maturity of a DevOps environment in any organization?***

To answer to this questions several steps were followed as described in section contributions. The first step was the SLR performed, which was essential for the identification of the processes, then the identified DevOps processes were the foundation com perform PRM and consequently PAM. Finally, the DevOps maturity model was created based on the previous step and as a cornerstone ISO/IEC standard as shown in figure 5.2.

## 8.2 Limitations

Regarding limitations, it was not possible to gather enough information and present a robust conclusion regarding specific topics, such as Outcomes, since DevOps is a recent subject. The current research cannot avoid biases since sources of literature written in other languages were excluded and certain unavailability to find

many studies addressing all DevOps related subjects in electronic databases. Since DevOps is recent, there are not a lot of experts in this area.

This research was meant to fill a gap that was found during the literature review(chapter 5). However, the resulting model does not achieve fully its purpose, thus it was not possible to do more than one interaction that could have provided more inputs to improve the existing outcomes and to the evaluation model.

The lack of literature addressing the DevOps processes made it very difficult to identify the base practices, inputs, and outputs of each process. We did our best to consult all results from the used search keys, however, some were inaccessible due to access restrictions. Certainly, it is not guaranteed that the chosen search keys, sufficiently represented the goal to retrieve all literature available on DevOps processes.

## 8.3 Future Work

This study has provided information about the DevOps maturity model and upon this creates a comprehensive model based on ISO/IEC. However, this research is not complete and can be taken a further look at. These opportunities will be described in this section.

The development of a tool that would allow the assessment to be made and that was flexible and adaptable to Companies to evaluate their maturity.

The DevOps maturity model created in this research can be used by other researchers to further build upon. This can be achieved by doing more research, which will not only contribute for DevOps maturity model but allow more researchers to use ISO/IEC a basis to these models.

Demonstrate and evaluate the framework to a substantially larger number of companies, ideally in companies that differ in size and industry, with the ultimate goal of being able to benchmark effectively.

# Bibliography


[1] Ken Peffers, Tuure Tuunanen, Marcus A Rothenberger, and Samir Chatterjee. A design science research methodology for information systems research. *Journal of management information systems*, 24(3):45–77, 2007.

[2] Barbara Kitchenham and Stuart Charters. Guidelines for performing systematic literature reviews in software engineering. 2007.

[3] Henner Gimpel, Sabiölla Hosseini, Rocco Xaver Richard Huber, Laura Probst, Maximilian Röglinger, and Ulrich Faisst. Structuring digital transformation: A framework of action fields and its application at zeiss. *J. Inf. Technol. Theory Appl.*, 19(1):3, 2018.

[4] Tobias Mettler and Peter Rohner. Situational maturity models as instrumental artifacts for organizational design. In *Proceedings of the 4th international conference on design science research in information systems and technology*, page 22. ACM, 2009.

[5] Gene Kim, Kevin Behr, and Kim Spafford. *The phoenix project: A novel about IT, DevOps, and helping your business win*. IT Revolution, 2014.

[6] Miguel Angelo Silva, João Pedro Faustino, Rúben Pereira, and Miguel Mira da Silva. Productivity gains of devops adoption in an it team: a case study. 2018.

[7] Rico de Feijter, Sietse Overbeek, Rob van Vliet, Erik Jagroep, and Sjaak Brinkkemper. Devops competences and maturity for software producing organizations. In *Enterprise, Business-Process and Information Systems Modeling*, pages 244–259. Springer, 2018.

[8] Barry W Boehm. A spiral model of software development and enhancement. *Computer*, (5):61–72, 1988.

[9] Adriana Herden, Pedro Porfirio Muniz Farias, and Adriano Bessa Albuquerque. An agile approach to improve process-oriented software development. In *Computer Science On-line Conference*, pages 413–424. Springer, 2016.

[10] G Bou Ghantous and Asif Gill. Devops: Concepts, practices, tools, benefits and challenges. *PACIS2017*, 2017.

[11] J.M. Radstaak. Developing a devops maturity model: A validated model to evaluate the maturity of devops in organizations. Master's thesis, University of Twente, 2019.

[12] Viral Gupta, Parmod Kumar Kapur, and Deepak Kumar. Modeling and measuring attributes influencing devops implementation in an enterprise using structural equation modeling. *Information and Software Technology*, 92:75–91, 2017.

[13] Jan Waller, Nils C Ehmke, and Wilhelm Hasselbring. Including performance benchmarks into continuous integration to enable devops. *ACM SIGSOFT Software Engineering Notes*, 40(2):1–4, 2015.

[14] Johannes Wettinger. Concepts for integrating devops methodologies with model-driven cloud management based on tosca. Master's thesis, University of Stuttgart, 2012.

[15] Avita Katal, Vinayak Bajoria, and Susheela Dahiya. Devops: Bridging the gap between development and operations. In *2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)*, pages 1–7. IEEE, 2019.

[16] Bob Aiello and Leslie Sachs. *Agile application lifecycle management: Using DevOps to drive process improvement*. Addison-Wesley Professional, 2016.

[17] Floris Erich, Chintan Amrit, and Maya Daneva. A mapping study on cooperation between information system development and operations. In *International Conference on Product-Focused Software Process Improvement*, pages 277–280. Springer, 2014.

[18] Jason Sharp and Jeffry Babb. Is information systems late to the party? the current state of devops research in the association for information systems elibrary. 2018.

[19] Roy Wendler. The maturity of maturity model research: A systematic mapping study. *Information and software technology*, 54(12):1317–1339, 2012.

[20] Lianping Chen. Continuous delivery: Huge benefits, but challenges too. *IEEE Software*, 32(2):50–54, 2015.

[21] Gerry Gerard Claps, Richard Berntsson Svensson, and Aybüke Aurum. On the journey to continuous deployment: Technical and social challenges along the way. *Information and Software technology*, 57:21–31, 2015.

[22] Wolfgang Gottesheim. Challenges, benefits and best practices of performance focused devops. In *Proceedings of the 4th International Workshop on Large-Scale Testing*, page 3. ACM, 2015.

[23] Mohammad Zarour, Norah Alhammad, Mamdouh Alenezi, and Khalid Alsarayrah. A research on devops maturity models. 8:4854, 2019.

[24] Lucy Ellen Lwakatare, Pasi Kuvaja, and Markku Oivo. An exploratory study of devops extending the dimensions of devops with practices. *ICSEA 2016*, 104, 2016.

[25] Pilar Rodríguez, Alireza Haghighatkhah, Lucy Ellen Lwakatare, Susanna Teppola, Tanja Suomalainen, Juho Eskeli, Teemu Karvonen, Pasi Kuvaja, June M Verner, and Markku Oivo. Continuous deployment of software intensive products and services: A systematic mapping study. *Journal of Systems and Software*, 123:263–291, 2017.

[26] Leonardo Leite, Carla Rocha, Fabio Kon, Dejan Milojicic, and Paulo Meirelles. A survey of devops concepts and challenges. *ACM Computing Surveys (CSUR)*, 52(6):1–35, 2019.

[27] Leah Riungu-Kalliosaari, Simo Mäkinen, Lucy Ellen Lwakatare, Juha Tiihonen, and Tomi Männistö. Devops adoption benefits and challenges in practice: a case study. In *17th International Conference on Product-Focused Software Process Improvement*, pages 590–597. Springer, 2016.

[28] Stephen Nelson-Smith. What is this devops thing, anyway. *Retrieved*, 3:2014, 2010.

[29] Manish Virmani. Understanding devops & bridging the gap from continuous integration to continuous delivery. In *Fifth International Conference on the Innovative Computing Technology (INTECH 2015)*, pages 78–82. IEEE, 2015.

[30] Mojtaba Shahin. Architecting for devops and continuous deployment. In *Proceedings of the ASWEC 2015 24th Australasian Software Engineering Conference*, pages 147–148. ACM, 2015.

[31] Michael Hüttermann. Gain fast feedback. In *DevOps for Developers Anonymous Spring*, pages 81–94. Springer, 2012.

[32] Eric Shamow. Devops at advance internet: How we got in the door. *IT Journal*, 14:14–16, 2011.

[33] Soon K Bang, Sam Chung, Young Choh, and Marc Dupuis. A grounded theory analysis of modern web applications: knowledge, skills, and abilities for devops. In *Proceedings of the 2nd annual conference on Research in information technology*, pages 61–62. ACM, 2013.

[34] Christiane Gresse von Wangenheim, Jean Carlo Rossa Hauck, Clenio F. Salviano, and Aldo von Wangenheim. Systematic literature review of software process capability / maturity models. 2010.

[35] Andreas Schumacher, Selim Erol, and Wilfried Sihn. A maturity model for assessing industry 4.0 readiness and maturity of manufacturing enterprises. *Procedia Cirp*, 52:161–166, 2016.

[36] Oliver Paulzen, Maria Doumi, Primoz Perc, and Anxo Cereijo-Roibas. A maturity model for quality improvement in knowledge management. *ACIS 2002 Proceedings*, page 5, 2002.

[37] B Curtis, W Hefley, and S Miller. People cmm: A framework for human capital management.(2nd), 2010.

[38] Mary Beth Chrissis, Mike Konrad, and Sandy Shrum. *CMMI guidlines for process integration and product improvement*. Addison-Wesley Longman Publishing Co., Inc., 2003.

[39] Alec Dorling. Spice: Software process improvement and capability determination. *Software Quality Journal*, 2(4):209–224, 1993.

[40] Roman Klimenko, Robert Winter, and Peter Rohner. Creating capability maturity model for agile transformation excellence.

[41] Lester Allan Lasrado, Ravi Vatrapu, and Kim Normann Andersen. Maturity models development in is research: a literature review. In *IRIS Selected Papers of the Information Systems Research Seminar in Scandinavia 2015. Paper*, volume 6, 2015.

[42] Marlies Van Steenbergen, Rik Bos, Sjaak Brinkkemper, Inge Van De Weerd, and Willem Bekkers. The design of focus area maturity models. In *International Conference on Design Science Research in Information Systems*, pages 317–332. Springer, 2010.

[43] Jos van Hillegersberg. The need for a maturity model for maturity modeling. In *The Art of Structuring*, pages 145–151. Springer, 2019.

[44] Jörg Becker, Ralf Knackstedt, and Jens Pöppelbuß. Developing maturity models for it management. *Business & Information Systems Engineering*, 1(3):213–222, 2009.

[45] Frank McGarry, Steve Burke, and Bill Decker. Measuring the impacts individual process maturity attributes have on software products. In *Proceedings Fifth International Software Metrics Symposium. Metrics (Cat. No. 98TB100262)*, pages 52–60. IEEE, 1998.

[46] James D. Herbsleb, Anita D. Carleton, James A. Rozum, Jane Siegel, and David Zubrow. Benefits of cmm-based software process improvement: Initial results. 1994.

[47] Dennis R. Goldenson and Diane L Gibson. Demonstrating the impact and benefits of cmmi ® : An update and preliminary results. 2003.

[48] D Gibson, D Goldenson, and Keith Kost. Performance results of cmmi-based process improvement. software engineering institute, 2006.

[49] ISO/IEC 33072. Information technology — process assessment — process capability assessment model for information security management, 2016.

[50] Tonia De Bruin, Ronald Freeze, Uday Kaulkarni, and Michael Rosemann. Understanding the main phases of developing a maturity assessment model. In B Campbell, J Underwood, and D Bunker, editors, *Australasian Conference on Information Systems (ACIS)*, pages 8–19, Australia, New South Wales, Sydney, 2005. Australasian Chapter of the Association for Information Systems.

[51] Mark C Paulk, Charles V Weber, Suzanne M Garcia, Mary Beth Chrissis, and Marilyn Bush. Key practices of the capability maturity model for software, version 1.1. *Pittsburgh, PA: Software Engineering Institute (SEI)*, 1993.

[52] Jane Webster and Richard T Watson. Analyzing the past to prepare for the future: Writing a literature review. *MIS quarterly*, pages xiii–xxiii, 2002.

[53] Lucy Ellen Lwakatare, Pasi Kuvaja, and Markku Oivo. Dimensions of devops. In *International conference on agile software development*, pages 212–217. Springer, 2015.

[54] Welder Pinheiro Luz, Gustavo Pinto, and Rodrigo Bonifácio. Adopting devops in the real world: A theory, a model, and a case study. *Journal of Systems and Software*, 157:110384, 2019.

[55] Welder Pinheiro Luz, Gustavo Pinto, and Rodrigo Bonifácio. Building a collaborative culture: a grounded theory of well succeeded devops adoption in practice. In *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 1–10, 2018.

[56] Antonio Capizzi, Salvatore Distefano, and Manuel Mazzara. From devops to devdataops: Data management in devops processes. In *International Workshop on Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment*, pages 52–62. Springer, 2019.

[57] Asif Qumer Gill, Abhishek Loumish, Isha Riyat, and Sungyoup Han. Devops for information management systems. *VINE Journal of Information and Knowledge Management Systems*, 2018.

[58] Pia Arentoft Nielsen, Till J Winkler, and Jacob Nørbjerg. Closing the it development-operations gap: The devops knowledge sharing framework. In *BIR Workshops*, 2017.

[59] Ramtin Jabbari, Nauman bin Ali, Kai Petersen, and Binish Tanveer. What is devops? a systematic mapping study on definitions and practices. In *Proceedings of the Scientific Workshop Proceedings of XP2016*, pages 1–11, 2016.

[60] Anna Wiedemann, Nicole Forsgren, Manuel Wiesche, Heiko Gewald, and Helmut Krcmar. Research for practice: the devops phenomenon. *Communications of the ACM*, 62(8):44–49, 2019.

[61] Daniel Stahl, Torvald Martensson, and Jan Bosch. Continuous practices and devops: beyond the buzz, what does it all mean? In *2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 440–448. IEEE, 2017.

[62] Breno B Nicolau de França, Helvio Jeronimo, and Guilherme Horta Travassos. Characterizing devops by hearing multiple voices. In *Proceedings of the 30th Brazilian symposium on software engineering*, pages 53–62, 2016.

[63] Mamdouh Alenezi, Mohammad Zarour, and Sultan Alsulis. Devops development process awareness and adoption-the case of saudi arabia. *i-Manager's Journal on Software Engineering*, 14(1):21, 2019.

[64] Ahmed Bahaa Farid, Yehia Mostafa Helmy, and Mahmoud Mohamed Bahloul. Enhancing lean software development by using devops practices. *International Journal of Advanced Computer Science and Applications*, 8(7):267–277, 2017.

[65] Bassam S Farroha and Deborah L Farroha. A framework for managing mission needs, compliance, and trust in the devops environment. In *2014 IEEE Military Communications Conference*, pages 288–293. IEEE, 2014.

[66] Nikhil Rathod and Anil Surve. Test orchestration a framework for continuous integration and continuous deployment. In *2015 international conference on pervasive computing (ICPC)*, pages 1–5. IEEE, 2015.

[67] Johannes Wettinger, Vasilios Andrikopoulos, and Frank Leymann. Automated capturing and systematic usage of devops knowledge for cloud applications. In *2015 IEEE International Conference on Cloud Engineering*, pages 60–65. IEEE, 2015.

[68] Jiabin Zheng, Yan Liu, and Jin Lin. Exploring devops for data analytical system with essential demands elicitation. In *SEKE*, pages 255–260, 2016.

[69] Floris Erich, Chintan Amrit, and Maya Daneva. Report: Devops literature review. *University of Twente, Tech. Rep*, 2014.

[70] Akond Ashfaque Ur Rahman and Laurie Williams. Software security in devops: Synthesizing practitioners' perceptions and practices. In *2016 IEEE/ACM International Workshop on Continuous Software Evolution and Delivery (CSED)*, pages 70–76. IEEE, 2016.

[71] Brian Fitzgerald and Klaas-Jan Stol. Continuous software engineering and beyond: trends and challenges. In *Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering*, pages 1–9, 2014.

[72] Brian Fitzgerald and Klaas-Jan Stol. Continuous software engineering: A roadmap and agenda. *Journal of Systems and Software*, 123:176–189, 2017.

[73] Johannes Wettinger, Uwe Breitenbücher, and Frank Leymann. Devopslang–bridging the gap between development and operations. In *European Conference on Service-Oriented and Cloud Computing*, pages 108–122. Springer, 2014.

[74] James Roche. Adopting devops practices in quality assurance. *Communications of the ACM*, 56(11):38–43, 2013.

[75] A. Agarwal, S. Gupta, and T. Choudhury. Continuous and integrated software development using devops. *2018 International Conference on Advances in Computing and Communication Engineering (ICACCE)*, pages 290–293, 2018.

[76] Nicolás Paez. Versioning strategy for devops implementations. In *2018 Congreso Argentino de Ciencias de La Informática y Desarrollos de Investigación (CACIDI)*, pages 1–6. IEEE, 2018.

[77] Zia Babar, Alexei Lapouchnian, and Eric Yu. Modeling devops deployment choices using process architecture design dimensions. In *IFIP Working Conference on The Practice of Enterprise Modeling*, pages 322–337. Springer, 2015.

[78] Noureddine Kerzazi and Bram Adams. Who needs release and devops engineers, and why? In *Proceedings of the International Workshop on Continuous Software Evolution and Delivery*, pages 77–83, 2016.

[79] Jan Pries-Heje, Richard Baskerville, and John R Venable. Strategies for design science research evaluation. 2008.

[80] John Venable, Jan Pries-Heje, and Richard Baskerville. Feds: a framework for evaluation in design science research. *European journal of information systems*, 25(1):77–89, 2016.

# DevOps *Practices*

| **Pratices** | [59] | [53] | [12] | [61] | [24] | [62] | [10] | [6] | [63] | [64] | [75] | [76] | [71] | [72] | [73] | [74] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Continuous Delivery | x | x | x | x | | x | | | x | x | x | x | | | x | |
| Continuous Integration | x | x | x | x | | | x | x | | x | | | x | x | | |
| Continuous Deployment | x | x | x | x | | x | | | x | | x | | | | x | |
| Continuous Monitoring | x | x | x | | | | | x | | x | | | | | | |
| Continuous Testing | x | x | x | | | | | x | | x | | | | | | |
| Infrastructure as a code | x | | | | x | | | | x | | | x | | | | |
| Automated software Testing | | | | | | x | | | x | | | | | | | |
| Continuous Release | | | | x | | | x | | | | | | | | | |
| Deployment Process Automation | | x | | | | | x | | | | | | | | | |
| Feedback Loop between developers and operators | x | | | | | | | | x | | | | | | | |
| Monitoring | | | | | x | | x | | | | | | | | | |
| Continuous Planning | x | | | | | | | | | x | | | | | | |
| Process Standardization | x | | | | | | | | | | | | | | | x |
| Application Monitoring | x | | | | | | | | | | | | | | | |
| Automated Dashboard | x | | | | | | | | | | | | | | | |
| Automated Configuration Management | | | | | | x | | | | | | | | | | |
| Automated Release | | | | | | | x | | | | | | | | | |
| Code Deployment | | | | | | | x | | | | | | | | | |
| Collaborative Development | | | | | | | | x | | | | | | | | |
| Continuous Business Planning | | | | | | | | x | | | | | | | | |
| Continues in the next page | | | | | | | | | | | | | | | | |

| **Pratices** | [59] | [53] | [12] | [61] | [24] | [62] | [10] | [6] | [63] | [64] | [75] | [76] | [71] | [72] | [73] | [74] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Continuous Customer Feedback and Optimization | | | | | | | | x | | | | | | | | |
| Continuous Collaboration | | | | | | | x | | | | | | | | | |
| Continuous Feedback | | | | | | | | | | x | | | | | | |
| Continuous Software Delivery | | | | | | x | | | | | | | | | | |
| Deployment Pipeline | | | | | | x | | | | | | | | | | |
| Improved Communication and Collaboration | | | | | | | | | x | | | | | | | |
| Increase the scope of responsibilities | | | | | x | | | | | | | | | | | |
| Infrastructure Configuration Management | | | | | | x | | | | | | | | | | |
| Prototyping Application | x | | | | | | | | | | | | | | | |
| Stakeholders participation | x | | | | | | | | | | | | | | | |
| Software Configuration Management | | | | | | x | | | | | | | | | | |
| Version Control | | | | | | | x | | | | | | | | | |
| Versioning | | | | | | | x | | | | | | | | | |

# B

# Included Studies

| Reference | Author | Title | Conference / Journal | Year |
|---|---|---|---|---|
| [60] | Wiedemann, Anna and Forsgren, Nicole and Wiesche, Manuel and Gewald, Heiko and Krcmar, Helmut | The DevOps phenomenon | ACM Queue | 2019 |
| [75] | Aayush Argarwal, Subhash Grupta and Tanupriya Choudhury | Continuous and Integrated Software Development using DevOps. | International Conference on Advances in Computing and Communication Engineering (ICACCE) | 2018 |
| [63] | Alenezi, Mamdouh and Zarour, Mohammad and Alsulis, Sultan | DevOps Development Process Awareness and Adoption - the Case of Saudi Arabia | i-Manager's Journal on Software Engineering | 2019 |
| [77] | Babar, Zia and Lapouchnian, Alexei and Yu, Eric | Modeling DevOps deployment choices using process architecture design dimensions. | IFIP Working Conference on the Practice of Enterprise Modelling | 2015 |
| [10] | Ghantous, Georges Bou and Gill, asif | DevOps: Concepts, Practices, Tools, Benefits and Challenges | Pacific Asia Conference on Information Systems (PACIS) | 2017 |
| [56] | Capizzi, Antonio and Distefano, Salvatore and Mazzara, Manuel | From DevOps to DevDataOps: Data Management in DevOps processes | International Workshop on Software Engineering Aspects of Continuous Development, New Paradigms of Software Production and Deployment | 2020 |
| [62] | de França, Breno B Nicolau and Jeronimo, Helvio and Travassos, Guilherme Horta | Characterizing DevOps by hearing multiple voices | Proceedings of the 30th Brazilian Symposium on Software Engineering | 2016 |
| [69] | F. Erich, C. Amrit, M. Daneva | Report: DevOps Literature Review | University of Twente | 2014 |
| [64] | Farid, Ahmed Bahaa and Helmy, Yehia Mostafa and Bahloul, Mahmoud Mohamed | Enhancing lean software development by using DevOps practices | International Journal of Advanced Computer Science and Applications | 2017 |
| [65] | Farroha, Bassam S and Farroha, Deborah L | A framework to manage mission needs, compliance, and trust in the DevOps environment | Military Communications Conference | 2014 |
| [71] | Fitzgerald, Brian and Stol, Klaas-Jan | Continuous Software Engineering and beyond: Trends and Challenges | International Workshop on Rapid Continuous Software Engineering | 2017 |
| [72] | Fitzgerald, Brian and Stol, Klass-Jan | Continuous software engineering: A roadmap and agenda | Journal of Systems and Software | 2017 |
| [57] | Qumer Gill, Asif and Loumish, Abhishek and Riyat, Isha and Han, Sungyoup | DevOps for information management systems | VINE Journal of Information and Knowledge Management Systems | 2018 |
| [12] | Gupta, Viral and Kapur, P.K. and Kumar, Deepak | Modeling and measuring attributes influencing DevOps implementation in an enterprise using structural equation modeling | Journal Information and Software Technology | 2017 |
| [59] | Jabbari, Ramtin and Ali, Nauman and Petersen, Kai | What is DevOps? A systematic mapping study on definitions and practices | Scientific Workshop Proceedings of XP2016 | 2016 |

continues next page

| Reference | Author | Title | Conference / Journal | Year |
|---|---|---|---|---|
| [26] | Leite, Leonardo and Rocha, Carla and Kon, Fabio and Milojicic, Dejan and Meirelles, Paulo | A Survey of DevOps Concepts and Challenges | ACM Computing Surveys | 2019 |
| [55] | Luz, Welder Pinheiro and Pinto, Gustavo and Bonifácio, Rodrigo | Building a collaborative culture: A grounded theory of well succeeded devops adoption in practice | Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement | 2018 |
| [54] | Luz, Welder Pinheiro and Pinto, Gustavo and Bonifácio, Rodrigo | Adopting DevOps in the real world: A theory, a model, and a case study | Journal of Systems and Software | 2019 |
| [53] | Lwakatare, Lucy Ellen and Kuvaja, Pasi and Oivo, Markku | Dimensions of DevOps | International conference on Agile Software Development | 2015 |
| [24] | Lwakatare, Lucy Ellen and Kuvaja, Pasi and Oivo, Markku | An Exploratory Study of DevOps Extending the Dimensions of DevOps with Practices | International Conference on Software Engineering Advances (ICSEA) | 2016 |
| [58] | Nielsen, Pia Arentoft and Winkler, Till J. and Nørbjerg, Jacob | Closing the IT development-operations gap: The devops knowledge sharing framework | Bibliometric-enhanced Information Retrieval (BIR) Workshops | 2017 |
| [76] | N. Paez | Versioning Strategy for DevOps Implementations | Congreso Argentino de Ciencias de la Informática y Desarrollos de Investigación (CACIDI) | 2018 |
| [70] | Rahman, Akond Ashfaque Ur and Williams, Laurie | Software Security in DevOps: Synthesizing Practitioners Perceptions and Practices | International Workshop on Continuous Software Evolution and Delivery (CSED) | 2016 |
| [66] | Rathod, Nikhil and Surve, Anil | Test Orchestration a framework for Continuous Integration and Continuous Deployment | International Conference on Pervasive Computing (ICPC) | 2015 |
| [74] | James Roche | Adopting DevOps Practices in Quality Assurance. | Communications of the ACM | 2013 |
| [6] | Silva, Miguel Angelo and Faustino, João Pedro and Pereira, Rúben and Mira da Silva, Miguel | Productivity Gains of DevOps Adoption in an IT Team: A Case Study | International Conference on Information Systems Development (ISD) | 2018 |
| [61] | Stahl, Daniel and Martensson, Torvald and Bosch, Jan | Continuous practices and devops: beyond the buzz, what does it all mean? | 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA) | 2017 |
| [73] | Johannes Wettinger, Uwe Breitenbücher, Frank Leymann | DevOpsSlang – Bridging the Gap between Development and Operations. | European Conference on Service-Oriented and Cloud Computing | 2014 |
| [67] | Wettinger, Johannes and Andrikopoulos, Vasilios and Leymann, Frank | Automated capturing and systematic usage of DevOps knowledge for cloud applications | International Conference on Cloud Engineering | 2015 |
| [68] | Zheng, Jiabin and Liu, Yan and Lin, Jin | Exploring DevOps for Data Analytical System with Essential Demands Elicitation | International Conference on Software Engineering and Knowledge Engineering (SEKE) | 2016 |