Master Thesis
for the study programme MSc. Business Information Technology

DEVELOPING A DEVOPS MATURITY MODEL

JEROEN M. RADSTAAK

A validated model to evaluate the maturity of DevOps in organizations

April 2019
University of Twente

Jeroen M. Radstaak: Developing a DevOps maturity model: A validated model to evaluate the maturity of DevOps in organizations
Master Thesis, *University of Twente* April 2019

Author
Jeroen M. Radstaak *(Study Programme: MSc. Business Information Technology)*
Study track: IT Management Innovation
j.m.radstaak@alumnus.utwente.nl

Graduation Committee
Dr. M. Daneva (*Faculty of EEMCS, University of Twente*
Prof Dr. Jos van Hillegersberg *(Faculty of BMS, University of Twente)*

In the past decade, practitioners and scholars have tried to agree on a definition for DevOps. Most of these definitions describe DevOps as an Agile methodology that combines the whole product lifecycle. This mainly includes but is not limited to, development and operation teams. One of the goals of DevOps is to create an environment in which short iterations of product releases can happen without stability issues. This can be achieved in multiple ways, however, the approaches to DevOps are commonly divided into five key dimensions: Culture, Communication and collaboration, Automation, Measurement, and Monitoring.

As the approaches to implementing DevOps can be vastly different is has been hard to establish a guideline for the implementation of such an environment. At the same time, many organizations struggle with implementing the soft aspects of DevOps and to convince other parties in the organization to get on board as well. Yet, collaboration and cooperation are needed for a successful DevOps implementation, especially regarding management support.

The goal of this study is to create a DevOps maturity model that will help in (1) guiding organizations to the required maturity level and in (2) demonstrating to other parties in the organization what is needed for a thriving DevOps environment. The DevOps maturity model is created based on a literature review and qualitative interviews with experts, followed by validation through expert opinions and put to practice in case studies.

The literature review revealed that multiple DevOps maturity models exist, both in practice-driven sources and scientific literature. However, our evaluation of the published models indicates that the models are either too broad or in an uncommon format and therefore difficult to use. It was chosen to combine the models into a new artefact that combines the common CMMI model with the detailed capabilities that are needed for a DevOps environment, which are found in the literature. This results in the first version of the DevOps maturity model (V1).

The second source of input is a group of five experts in the field of DevOps that work in Vietnamese companies and that participated in the qualitative interview study included in this research. These interviewees have different positions and backgrounds in multiple organizations which allowed them to create varied input for those capabilities that are contributing to a thriving DevOps environment. This input is combined with the first version of the DevOps maturity model and results in the DevOps maturity model V2.

The second version of the model was subjected to a first evaluation by means of an interview with the same interviewees that provided input for the model. The model was thoroughly discussed, and changes were made after each interview to improve the clarity, readability, and completeness of the model. This resulted in the final iteration of the DevOps maturity model, V3.

This final model was applied to real projects in several organizations during multiple case studies. Each case study consisted of an assessment that resulted in a completed DevOps maturity diagram, with a level of maturity selected for each

capability. This model should help to understand where the organization, and more specifically each project stands in terms of DevOps maturity.

The model has been created in multiple iterations and the practical application of the model has become clear during the case studies. However, the model is, just like DevOps itself, open to continuous improvement. For example, research on which levels of DevOps maturity are most common and could be considered the industry standard.

Research has shown that organizations had difficulties adapting to the CMMI model, because of cost, time and other models that were already used by the organization. Therefore, recommendations on how to apply the DevOps maturity model efficiently and description of the benefits have been made, to ensure the acceptance of the DevOps maturity model.

The research in this master thesis also contributes to empirical research in (South)East-Asia on DevOps in general. Previous publications have shown that the existing DevOps research is very scarce in this region[1]. Business cultures vary greatly across the world and it could be interesting to look closer at similarities and differences of the culture dimension. This comparison is currently hard to make as scientific literature on DevOps in (South)East-Asia is lacking.

This study has both scientific and practical implications:

- *scientific:* This study provides a new approach to DevOps maturity models and contributes to a limited amount of scientific research on DevOps maturity.
- *scientific:* Empirical research in (South)East-Asia on DevOps, in general, is very limited. This study contributes and can be used as a comparison to other geographical areas.
- *practical:* A common model, CMMI, has been adapted to fit a DevOps maturity assessment and should be able to be adapted easily inside an organization.
- *practical:* The model can be used to improve the knowledge on DevOps throughout the organization, by showing what is necessary to achieve different maturity levels

Limitations of the study are traceable to the fact that our research only took place in one geographical area and this could limit the usefulness of the model to organizations in different locations. However, this impact is limited because the origin of the model comes from existing literature. Another key limitation is the limited information on the projects that were assessed in the case studies. The amount of information that could be shared by the participating organizations was mostly limited to the amount of people, age of the project and number of teams involved. This also limited the discussion of the results. Future research can be done by doing more empirical research on DevOps maturity to further improve upon the DevOps maturity model.

1 This research was performed by the same author but has not been published. It can be requested by contacting the author.

ACKNOWLEDGEMENTS

This thesis is the conclusion to five and a half years of studying the business information technology BSc. And MSc. at the University of Twente. It has been an amazing time of personal growth, both inside and outside of the university. This thesis has taken me 6 months to complete, however I would not have been able to manage this without the help of other people.

First and foremost, my 1st supervisor Maya Daneva. She tirelessly provided me with feedback and support and nudged me into the right directions throughout this thesis adventure. Also, my 2nd supervisor, Jos van Hillegersberg, provided me with critical feedback that made this thesis into what is now in front of you.

I would also like to thank the organizations that have been supporting and willing to spend time doing interviews with me on multiple occasions. The enthusiasm during these interviews is what reminded me for being on the right track and that I was creating a useful tool that these people will be able to use. Also, special thanks goes out to the people that helped me get the connections with the right people at the right organizations.

Finally, I would like to thank my partner who has been with me through all the fun and difficulties I encountered throughout the process of writing my thesis and the adventure that living in Vietnam was. Also, my parents for supporting me in many ways throughout my studies.

CONTENTS

LIST OF TABLES

ACRONYMS

AWS	Amazon Web Services
CMMI	Capability Maturity Model Integration
IAC	Infrastructure as Code
QA	Quality Assurance
SAAS	Software as a Service
KPA	Key Process Areas

1 INTRODUCTION

In the past decade DevOps, a new approach in the agile software development movement combining development and operations, has gained popularity and is increasingly used in the software engineering community [4]. DevOps does not only include the development teams, as for example scrum methodology does, but also includes the operations teams of the software product development. A need for this started to exist after development iterations were getting shorter and more automation of the operations side was needed to keep up with development. These operations include, but are not limited to, automated testing, deployment, and monitoring.

1.1 PROBLEM STATEMENT

The approach to DevOps has strongly developed over the past decade into an approach which is now widely used in software engineering companies. This results in DevOps being an integral part of the delivery process of new software. It raises the question about the point of time at which DevOps is considered by a company to be fully implemented and the organizational characteristics that tell practitioners that the level of full implementation has in fact been reached. Maturity models have been used in software engineering to, for example, assess the capabilities (Capability Maturity Model) or the business intelligence in a company. A maturity model for DevOps could therefore be a useful metric to analyze the approach and techniques followed by any organization regarding DevOps. This results in the following research goal for this study:

Develop a suitable DevOps Maturity framework for assessing the maturity of a DevOps environment in any organization.

To reach this goal it will be necessary to follow a design science methodology, which is further detailed in chapter 3.

1.2 THESIS STRUCTURE

This master thesis adheres to the following structure:

- *Chapter 2* provides the background for this research, first about DevOps in general and followed by background on maturity models.
- *Chapter 3* describes the design of this research and the steps that will be taken to complete it
- *Chapter 4* consists of the literature review that results in the first iteration of the DevOps maturity model
- *Chapter 5* details the interview rounds with the experts. The first round to acquire more capabilities to add to the model and the second round for validation of the model, resulting in v2 and v3 of the DevOps maturity model respectively.
- *Chapter 6* describes the case studies that were performed and the results of the DevOps maturity assessment in relation to the model.
- *Chapter 7* provides organizations insight in how to use the DevOps maturity model
- *Chapter 8* discusses the implications of this research
- *Chapter 9* summarizes and concludes this report.

2 BACKGROUND

2.1 DEFINITIONS OF 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 [5]. Even though this is more than a decade ago it seems that research on DevOps is still in its infancy [6]. Previous literature reviews have shown that no set definition of DevOps exists [7]–[9]. This section will provide more insight into how recent studies have tried to define DevOps by going into more detail on the definition.

Recent studies describe DevOps as vaguely defined and loosely used in the software engineering community [4]. 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 [10] by facilitating a lean connection between development, delivery, and operation [11]. This is supported by practices like continuous testing and deployment [10]. What seems to be most common in the interpretation of DevOps is, that it is a culture shift needed toward the collaboration between development, quality assurance, operations and any other team involved with delivering software [11], [12]

According to Humble and Molesky DevOps can be achieved through four main principles: culture, automation, measurement and sharing [12]. These principles are also mentioned by Lwakatare et al. who also adds monitoring as well as collaboration and communication [13], [14]. Sharing is considered a part of the collaboration and culture as well. Five of these dimensions, sharing excluded to prevent redundancy, will shortly be discussed in the following section.

2.1.1 Collaboration and communication

The basic aspect of collaboration in DevOps is the implied collaboration between the Development and Operation side of software engineering. The literature research of 2017 by Ghantous and Gill researched this more in-depth and showed that collaboration and communication is the most frequently mentioned conceptual element to describe DevOps [9]. Lwakatare splits this up in two main practices [13]: “Increasing the scope of responsibilities" and “intensifying cooperation and involvement in each other daily work."

2.1.2 Automation

Automation is necessary to achieve the assumptions made by Humble in regards to DevOps: “achieving both frequent, reliable deployments and a stable production environment" [12]. This can be achieved by creating a continuous delivery process which consists of continuous planning, integration, deployment, testing,

and monitoring [15]. For example, automatic deployment is the ability to quickly deploy new releases into a production environment [16]. This automation can be supported by using an IT setup named infrastructure as code (IaC) [13].

The IaC concept is based on the idea that the entire infrastructure provisioning should be maintained as code in a source code repository [15]. 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 [17].

2.1.3 Culture

Transitioning to a new culture can be difficult, however, Shamow described the focus on change of culture in companies a necessity for adopting DevOps [18]. 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 [13]. An important step is to be involved in each other's work. This can be achieved 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 means that also developers should be on call for production issues [12].

To foster 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 [19]. Organizations should give employees trust in making the right decisions necessary for implementing automation of the process by creating a culture of personal responsibility [20].

2.1.4 Monitoring

Monitoring is multipurpose, it can be used to detect problems early on but also 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 [13].

2.1.5 Measurement

Measurement is important for evaluating the success of both the development and operation teams. This can be achieved with high-level and low-level metrics.

For example, 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 behavior, for example, the impact of product releases on the stability of the affected systems [12]. Using similar metrics ensures that both teams are trying to achieve the same goal, a stable production environment.

2.2 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 [21]. The perceived benefits of maturity models resulted in the creation of a wide range of software process capability and maturity models.

Most maturity models can be categorized as a fixed level maturity model or a focus area maturity model [22]. The fixed maturity level model has a fixed number of maturity levels, often following the maturity levels as defined in CMM [23]: 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 a more gradual a varying improvement path by allowing KPAs to be scored at different levels [24]. Another type of maturity model consists of focus 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 [25].

A procedure model for developing maturity models is described by Becker et al. [26]. 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. However, unfortunately, many maturity models miss the empirical evidence that could reveal the validity and usefulness of the models [27]. Resulting in maturity models that have an unclear sense of quality because of the lack of structure during the development of the models. A maturity model for developing maturity models may be needed to further guide the development of future maturity models [24].

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 environments has been shown by multiple researchers through empirical research [28]–[31]. 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 on investment as described by Goldenson and Gibson based on their research in 35 organizations [32], [33].

The CMMI model and its adaptation will be further detailed in Section 2.2.1. Also, in recent years the increased usage of Agile methodology has resulted in combinations between CMMI and agile, which is described in section 2.2.2.

2.2.1 CMMI model and its adaptations

As many maturity models are adapted from the CMMI framework it is necessary to take a closer look at this framework [21]. The CMMI emerged in the start of 1990s in the area of software engineering to improve software development processes to achieve higher quality and has since been used by hundreds of organizations worldwide [27]. The basics of the CMMI are high-level and consists of five levels of maturity. Starting with 'initial', in which processes are still unpredictable, poorly controlled and processes are reactive. The second level is 'managed', processes are focused on projects and are still often reactive. Level 3 is 'defined' and the processes are now characterized for the organization and not reactive, but proactive. Next is level 4 'quantitatively managed' in which processes are measured and controlled. Finally level 5 is 'optimizing', focusing on process improvement [1]. These levels are shown visually in figure 2.1.

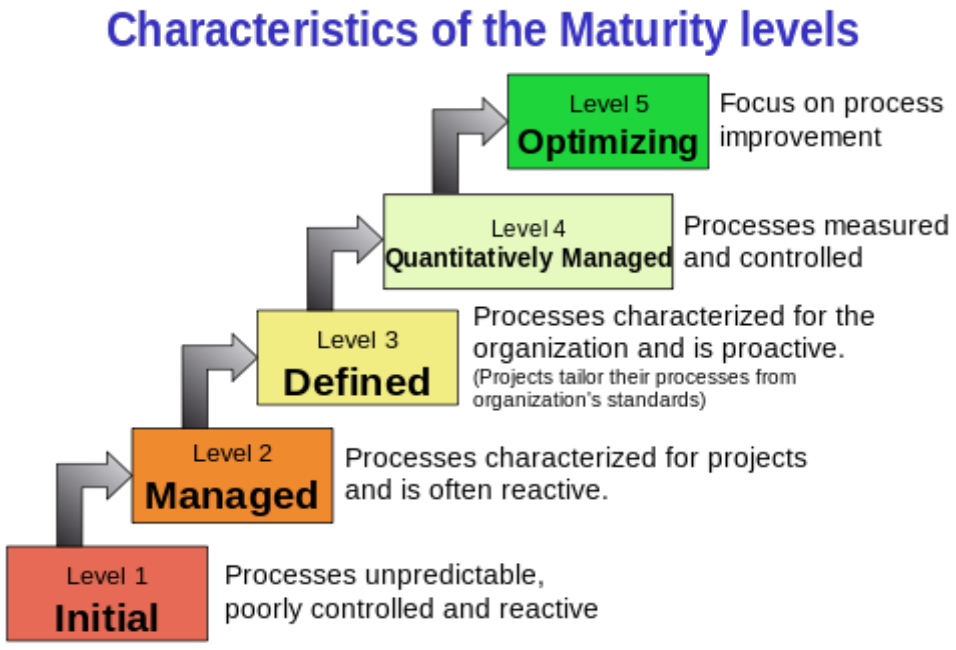

Figure 2.1: CMMI process model [1]

2.2.2 *Agile and CMMI*

The success of CMMI inspired several other maturity models in other domains, among others, also in software engineering focused on improve software development processes to achieve higher quality [27]. As DevOps is a new approach in the agile software development movement, it is necessary to look at the influence of Agile on CMMI.

Early research questioned the combination of Agile and CMMI. Developers using CMMI were developing for large-scale, risk-averse and mission-critical systems, with high levels of management oversight and hierarchical governance. Whereas Agile methods focus on smaller, single-team development projects with volatile requirements. It, therefore, seems that both are on opposite ends of each other [34].

However, more recent literature shows that the combined approach of agile and maturity models brings potential benefits and combined with tool support it can help increase the success rate of these initiatives [35]. Selleri et al. confirms, in a state-of-research review on the use of CMMI in adaptive software development, that combining CMMI and agile methods have allowed companies to achieve the levels 2 and 3, 'managed' and 'defined' respectively, with little effort [36].

3

RESEARCH DESIGN

This chapter describes the research objective and its supporting questions. Furthermore, the research approach will be described. The combination of these two will result in the final research methodology that will be followed step by step in this research

3.1 RESEARCH OBJECTIVE AND QUESTIONS

This research sets out to reach the following research objective (RO):

RO. What is a suitable maturity framework that allows organizations to assess and improve their DevOps environment?

In order to design an effective and comprehensive DevOps maturity model, it is necessary to research the design context. This prevents that everything must be made from scratch and that this research can be a continuation from existing research. The first sub-question (SQ) is defined accordingly.

SQ1. What models on maturity in DevOps exist and which are suitable to use for creating a comprehensive model applicable to all businesses working with DevOps?

The answers to SQ1 will enable us to create a first iteration of the maturity model. This maturity model will be used in SQ2. This question focuses on improving the model further by gathering input from experts in the DevOps field. The reason that experts are interviewed is that the goal is to create a model that can be used in practice by organizations to assess their maturity and improve their DevOps environment.

SQ2. What do experts in the DevOps field see as the capabilities necessary for successful DevOps implementation?

The results of SQ2 will result in the second iteration of the DevOps maturity model. This model will be validated by the same experts in the field to ensure that the model will fit the practical use that is envisioned.

SQ3. How do experts in the DevOps field evaluate the maturity model and what improvements should be done based on their responses?

The suggestions from SQ3 will result in version 3 and the final iteration of the DevOps maturity model. This model will be applied in practice and that is what SQ4 will focus on; testing the applicability of the DevOps maturity model.

SQ4. What is the DevOps maturity level of the organizations used for the case studies and how can this be explained?

SQ4 will result in maturity assessments of the projects in the organizations that participate in the case studies. This will be helpful for the organizations, but it can also be used as input for further improvement of the DevOps maturity model. This is what SQ5 focuses on, any improvements arising from the case studies

will be evaluated and suggested for further iterations of the DevOps maturity model.

SQ5. How did the DevOps maturity model perform in this case study and what improvements should be done based on these findings?

The result of answering all five sub-questions will be a comprehensive understanding of the context of DevOps maturity models and the creation of a DevOps maturity model for assessing the maturity of DevOps in organizations and thus giving the means to satisfy the research objective. Finally, suggestions for further improvements to the maturity model will be given as input for future research that will build on this research.

3.2 RESEARCH DESIGN

The design of the DevOps maturity model is structured according to the activities of the Design Science Research Methodology of Peffers et al. as shown in figure 3.1 [37]

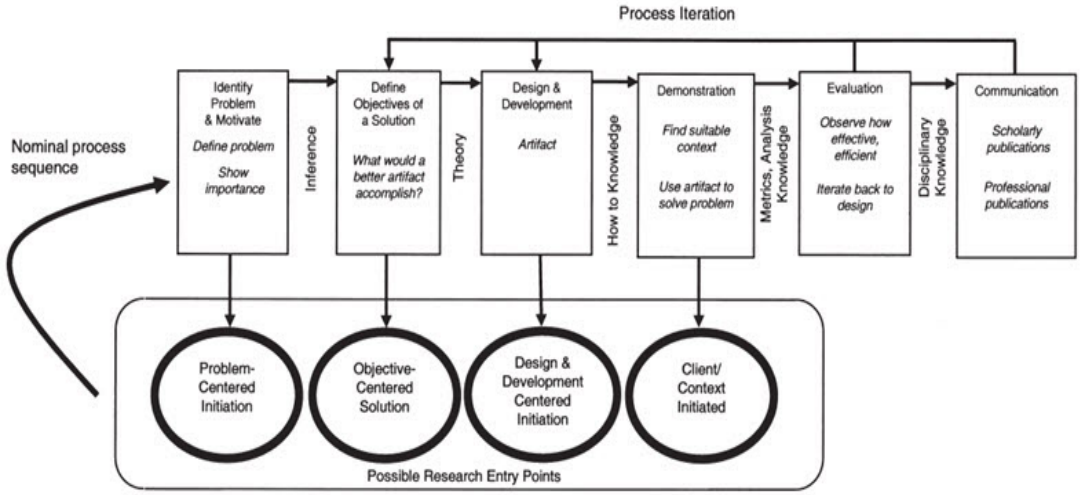

Figure 3.1: The DSRM Process Model

Identify Problem and Motivate
Two problems can be identified to motivate this literature. First, in the problem statement in chapter 1.1, it became clear that a need exists for a DevOps maturity model. Secondly, a need exists for more empirical research on DevOps in general in (South)East-Asia, because culture is a core component of DevOps and the culture in organizations differ between geographical regions[1] [38]. This possible need for different development methodologies, because of cultural differences is confirmed by Alpar [39], who considers differences in culture to be a core issue for high employee turnover (which relates to the capabilities of team organization and collaboration in DevOps) in IT outsourcing organizations in Vietnam.

1 This research was performed by the same author but has not been published. It can be requested by contacting the author.

Define Objectives and Solution
A DevOps maturity model will be created to resolve the identified problems. The solution will consist of a literature review, expert interviews and expert validation for the creation of a DevOps maturity model.

Design and Development
The design of the model consists of three iterations. The first version will be created from the analysis of existing DevOps maturity models. The second version will follow from combining the first version with interviews with experts on DevOps. The third, and final, version will follow by validation through expert opinion. The experts will imagine how the model would interact with the problem context and then predict the results. If these predictions are unsatisfactory, it would mean that the model must be redesigned.

Demonstration
The newly created DevOps maturity model is used in multiple case studies for measuring the maturity of the respective DevOps implementation. This will provide insights into the applicability of the model.

Evaluation
The results of the case studies will be analyzed, and suggestions for changes and future research regarding the DevOps maturity model will be given.

Communication
The results of this study are communication in two manners; this research report and as results of the maturity assessment for each organization that participated in the case studies.

3.3 RESEARCH METHODOLOGY

Combining the research objectives and questions with the research design leads to the model in figure 3.2. This model is similar to the process model of Becker et al. described in the Background section [26]. Data will be gathered through literature review and expert opinion. The design of the DevOps maturity framework therefore consists of three iterations. The first concept DevOps maturity framework is based on the literature review (SQ1). The second iteration will be with the additions resulting from the interviews with experts of the field (SQ2). These same experts will participate in a second interview round and will be asked to validate the DevOps maturity model v2 (SQ3). This validation round will result in the third and final iteration of the DevOps maturity model. This DevOps maturity model v3 will be applied in case studies to different projects (SQ4). The results will be discussed, and possible alterations will be suggested for future research (SQ5). These steps will result in answering the RO. This empirical approach was deemed necessary based on the lack of empirical research concerning DevOps maturity. A more detailed approach to the literature review, the two rounds of interviews with experts and, finally, the company case study will be described before each corresponding section, chapter 4, 5 and 6 respectively.

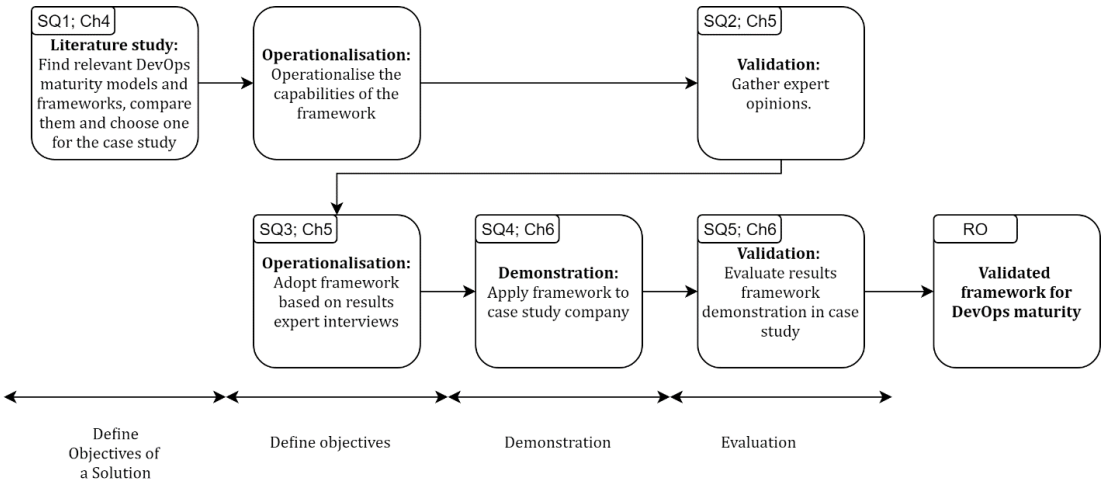

Figure 3.2: Design science methodology in the context of this research

4

LITERATURE REVIEW

As described in the Methodology section it was chosen to start with a literature review on existing DevOps maturity models to create a comprehensive overview of the context of DevOps maturity models. This first part of this section starts with the details of the literature review. This is followed by the analysis and discussion of the various models.

4.1 LITERATURE REVIEW METHOD

A structured literature review, implementing the practices of Kitchenham [40], was conducted to create a comprehensive overview of the existing DevOps maturity models from both research and practice. The databases used to retrieve scientific literature were Scopus, Web of Science, IEEE Xplore, ACM Digital Library, AIS Electronic library, Springer database and finally Google Scholar for exploratory purposes. For these databases the following search query was used:

"devops" AND ("CMMI" OR maturity OR "maturity model")

In order to assure the quality of literature only academic databases were used at first. However, this provided an unsatisfactory amount of results. Therefore, it was decided to also include "grey literature" (non-peer reviewed literature) in this research, which is especially important to do in the case of DevOps, because this field of research does not provide a substantial amount of literature [41]. A multivocal literature review, in which "white literature" and "grey literature" is combined, does require special attention to the quality of the papers. Therefore, the guidelines of Garousi, Felderer and Mäntylä were used on deciding if the "grey literature" should be included [42].

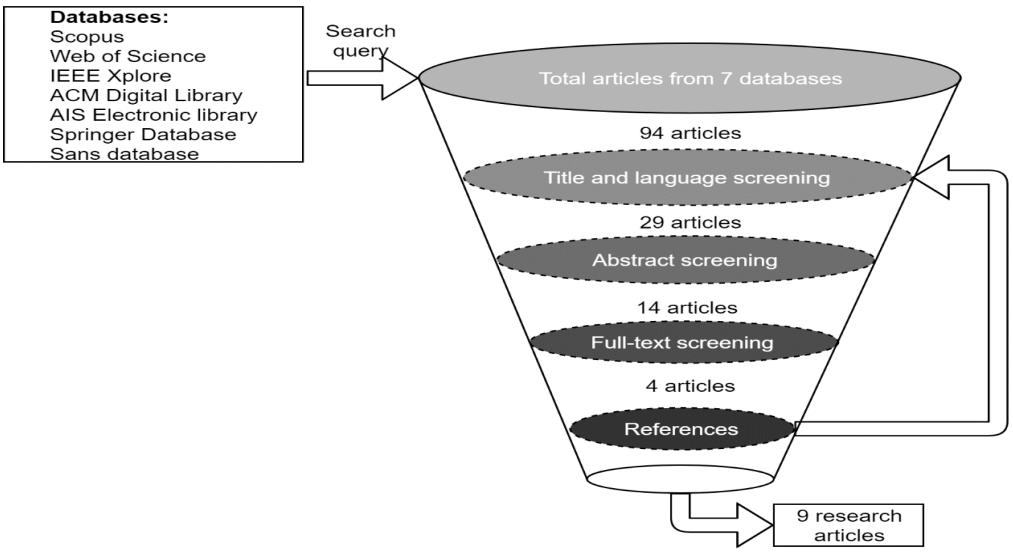

Figure 4.1: Literature review filtering process

The process of finding suitable literature including the search terms, exclusions, inclusions and summaries of the papers are shown in Appendix A. Our initial search yielded 94 papers. Using the inclusion and exclusion criteria on the title of the papers, 65 papers were excluded. Then, using the inclusion and exclusion criteria on the abstracts of the remaining 29 papers, we excluded further 15 publications. The remaining papers then were 14, which were read by using the whole text and only 4 were selected. The forward and backward references of these papers were checked for suitable literature. Five of those references were deemed suitable after a full-text screen and thus added to the selected literature. This brought the total of suitable research articles to nine. The gradual reduction of relevant papers is shown in Figure 4.1.

4.2 DEVOPS MATURITY MODELS FROM LITERATURE

The selected literature provides several maturity models, and some of these will be discussed in this section.

The first model is inherited from an existing model created by Hewlett Packard; however, it does not become clear from the literature if this model has been validated. Even though the validation of the model is unclear, it has been published in the Journal of Computer Engineering and been cited 10 times (February 2019) in recent literature. Therefore, it is chosen to describe this model as it can provide useful input for the creation of the DevOps maturity model in our research.

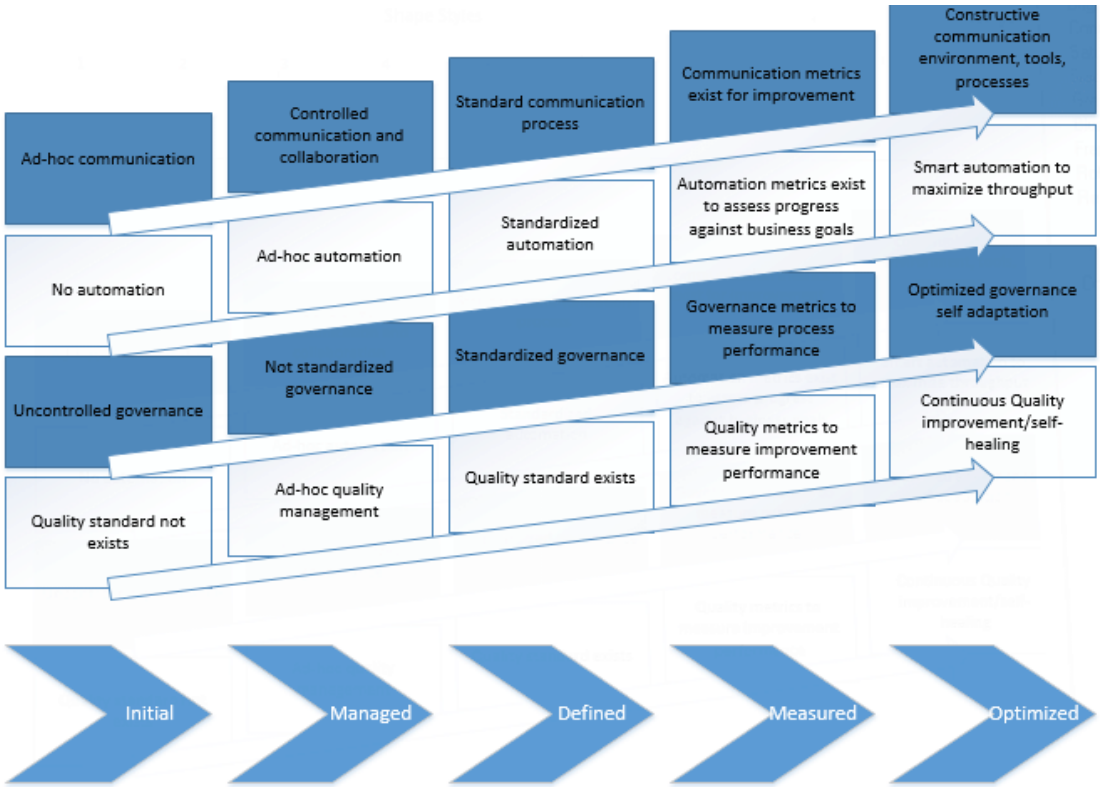

Figure 4.2: DevOps maturity model by Mohamed [43]

The proposed model by Mohamed [43] is a CMMI inspired model that was created by in 2015, shown in figure 4.1. In this model, each maturity level is set against 4 dimensions: communication/collaboration, automation, governance, and quality.

At the first level, there is only ad-hoc communication, no automation, uncontrolled governance, and no quality standard. By improving these it is possible to shift to the next level. Eventually reaching level 5, 'optimized'. This stage has the characteristics of, a collaboration between teams with a constructive communication environment, optimized automation to maximize throughput, self-adapting governance and continuous quality improvement [43].

More CMMI inspired maturity models existing, some from academic sources [44], but mainly as white papers by the companies Forester Consulting, IBM and InfoQ [45]–[48]. These models are adapted from CMMI, but all consist of similar approaches and levels as the maturity model by Mohamed. Therefore, it is chosen to only describe one of the white paper models to avoid redundancy.

Eric Minick created a DevOps maturity model at IBM in February 2014 [47]. It is unclear how this model was validated; however, it seems safe to assume that this model is used inside IBM and thus applied to a practical DevOps environment. The maturity model has four 'essential elements', building, deploying, testing and reporting. These four dimensions are divided into five different levels; base, beginner, intermediate, advanced and extreme. The base level is described as just doing enough "to be on the model" , whereas extreme is described as only being suitable for some teams but too expensive to achieve for most teams. Each of the elements has a level that is marked as the industry norm and one that is marked as a common target suitable for most teams. The industry norm and target of each essential element will be shortly described. An example of levels of the essential element 'building' is shown in figure 4.2.

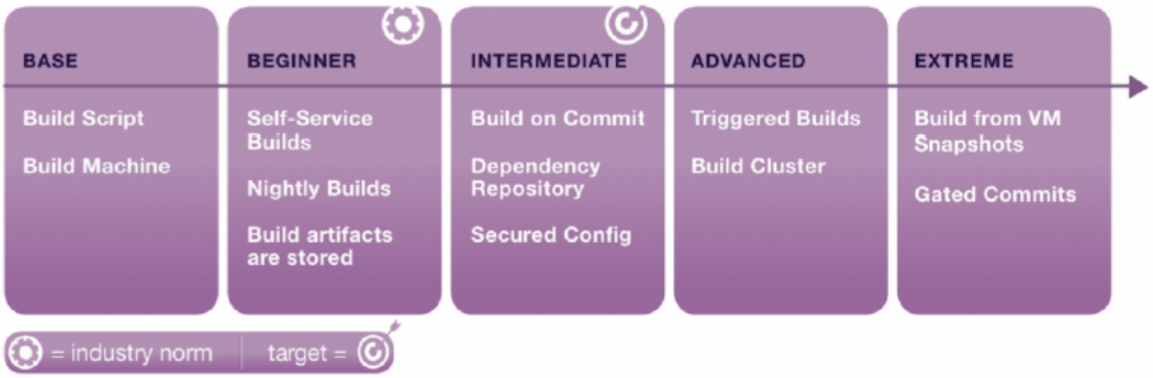

Figure 4.3: DevOps maturity levels for the element 'building' [47]

The base level requires to have a build machine and build script. The industry norm is the beginner level in which self-service build, nightly builds and build artifacts are stored. The target, suitable for most teams, is to have: a build whenever a developer commits code, dependency repository, tracing of all dependencies in-between software, and a secured build configuration.

For deploying the base level is the industry norm and consists of deployment scripts. However, the advanced level is the target fitting most teams. In this level, the deployments are fully automated, including database deployment, deployment onto some testing environments after passing quality gates. Also, interdependencies between deployments are mapped and considered.

The testing element has the industry norm at the base level with some test automation and the target at intermediate. Intermediate considering testing means that some static analysis occurs and that automated functional tests run nightly.

Finally, for reporting the beginner level, with a report visibility to the whole team and the latest reports always being accessible is the industry norm. The target is the advanced level which also takes trends between the reports into account.

Another approach to a maturity model for measuring DevOps maturity, shown in figure 4.3, was created by de Feijter et al. [2]. The model was created through data collection, consisting of semi-structured interviews and literature review, validated through two rounds of expert opinions and applied in an exploratory case study.

This model aims to not only measure DevOps maturity but to also provide guidance in adopting it in software producing organizations. This model consists of ten levels instead of the five in CMMI models and is more detailed. It was created from a capability model describing the capabilities necessary for DevOps adoption. This model can be split into three sections:

- Culture and collaboration
- Product, process and quality
- Foundation

Each section has its own focus areas, which will be described by their extremes to prevent a lengthy description. For example, the lowest level of 'communication' is 'indirect communication', communication between teams is established via managers and procedures, and the highest is 'communication improvement', which means adopting and trying out new communication practices from industry and learning from it. The lowest level of 'knowledge sharing' is 'decentralized knowledge sharing', knowledge is shared through notes and documents between professionals, and the highest level is 'communities of practice', knowledge is share between multidisciplinary professionals that share a common interest. When looking at 'trust and respect', the first level is 'culture of trust and respect imitation, in which professionals get tasks assigned from different teams to create trust and respect' while the highest level is 'culture of trust and respect shared core values', successes and rewards are shared while being transparent and open towards one another to prevent blaming. In respect to 'team organization' the initial level is 'separate teams' and the highest level is 'cross-functional teams with knowledge overlap', with T-shaped professionals having knowledge in both Development and Operations. Finally, 'release alignment' is the last focus area of culture and collaboration, the first level is 'roadmap alignment' and the highest

level is 'external release heartbeat alignment', that means that the release heartbeat is aligned with its dependent external stakeholders.

Focus area \ Level	0	1	2	3	4	5	6	7	8	9	10
Culture and collaboration											
Communication		A				B	C			D	E
Knowledge sharing				A		B	C				D
Trust and respect							A	B	C		
Team organization		A	B						C	D	
Release alignment				A					B	C	
Product, Process and Quality											
Release heartbeat		A				B	C		D	E	F
Branch and merge			A	B		C		D			
Build automation			A	B		C					
Development quality improvement			A			B		C	D	E	
Test automation				A	B	C			D		E
Deployment automation					A	B		C			D
Release for production					A			B	C	D	
Incident handling			A					B	C	D	
Foundation											
Configuration management			A	B		C					
Architecture alignment			A					B			
Infrastructure				A			B	C	D		

Figure 4.4: DevOps focus area maturity model [2]

The next section starts with the 'release heartbeat', the lowest level is described as 'requirements and incidents gathering and prioritization', requirements and incidents are gathered and prioritized with stakeholders, the highest level is 'release heartbeat improvement', with continuous improvement of the value stream by identifying and eliminating activities that do not add value and shortening of lead times and feedback moments with the customer. Next up is 'branch and merge', the first level is 'version-controlled source code' and the highest is 'feature toggles', in which functionality can be toggled in releases for customers. Followed by 'build automation', the lowest level is 'manual build creation' while 'continuous build creation' is the highest level, for which a build is created after each check-in of new code. The next one is 'development quality improvement' ranging from 'manual code quality monitoring', such as code reviews and adherence to code conventions, to 'quality gates', having quality gated check-ins with quality of code measurements. The next focus area is 'test automation', with the first level 'systematic testing', having manual unit and acceptance tests are performed systematically, and the highest level 'automated recoverability and resilience testing', providing randomly performed resilience tests in production and automated recoverability. The following focus area is 'deployment automation', for which the first level is 'manual deployment' and the highest level 'continuous deployment', with each code check-in automatically deployed into production. Next is the 'release for production' focus area ranging from 'definition of done', incorporating development and testing criteria to be complied with during a sprint to 'automated material generation', supporting materials automatically

being generated. The last focus area of this section is 'incident handling', ranging from 'reactive incident handling' to 'automated root cause detection', supported by analytics.

The final section starts with 'configuration management' ranging from 'manual configuration management', versions of used software are registered manually, to 'version-controlled configuration management', the versions of configurations are maintained in a version control. Followed by 'architecture alignment' consisting of the lowest level 'software and technical architecture alignment' and the highest level 'continuous architecture evolvement', with co-evolving of the software and technical architecture. The last focus area is 'infrastructure', its first level is 'manually provisioned infrastructure', provisioning all the infrastructure manually, and the last level, 'managed platform services', such as preconfigured platforms with possibility for automatic deployment of infrastructure. This concludes the description of the model by Feijter et al.

Finally, in the paper by Nagarajan and Overbeek [49] it is chosen to not create a typical DevOps maturity model, but a framework with specific capabilities for large agile-based financial organizations. This framework was created by means of a literature review and practitioner interviews and validated through expert opinions. The framework differs from other maturity models, however, it still serves the same purpose as the other maturity models described, as it can be used to measure and achieve DevOps transformation goals.

4.3 DISCUSSION

The models discussed in the previous sections approached DevOps maturity models differently. However, the definitions of the capabilities of each model have similarities. This can be seen in the comparison of the capabilities of the literature, which is shown in table 4.1. All capabilities that were only mentioned by one article were removed from the overview because those will not be used in the DevOps maturity model. This is done to prevent adding capabilities that are not sufficiently validated. However, if those excluded capabilities present itself in the upcoming design stages then they will be added to the next iteration of the maturity model. Also, some names or descriptions of capabilities differed only slightly and therefore have been grouped together under the most descriptive name. The full table, including the capabilities that were only mentioned once, is shown in Appendix B

The model that will follow from these capabilities is based on the seven pieces of literature, however, that does not necessarily mean that the list is complete. It is important to validate the model that follows from this literature review. In this case that will be done by expert interviews regarding the model as well as case studies on the applicability of the model. This should make the model more robust and useable in other organizations or situations.

Finally, each model that was presented in the literature had some questioning of the highest level and if it would be necessary to reach this level of DevOps maturity. It could turn out to be very expensive to implement all the capabilities

needed for the highest level. It, therefore, depends on the organization what level of maturity is needed. As described by Minick: “Most organizations would be crazy to implement them, while a minority would be crazy to not implement them" [47].

Also, worth noting was a case study was carried out on a real-world project by Rong et al. to explore the feasibility of applying CMMI models to guide improvements for DevOps projects [50]. It was discovered that CMMI is not a direct fit for DevOps projects. One of the issues found was with respect to the usability in conducting appraisals, as the CMMI model is not able to keep up with quick iterations in DevOps. Nevertheless, the CMMI framework could provide a good foundation for creating an extended version that would fit DevOps projects.

Table 4.1: Comparison of literature on DevOps capabilities

Capabilities	**[43]**	**[44]**	**[45]**	**[46]**	**[47]**	**[2]**	**[49]**
Constructive communication environment	X					X	X
Manage environments through automation		X	X				
Continuous process improvement		X					X
Create cross-functional teams with shared KPI and knowledge overlap		X		X		X	X
Allow rapid personnel feedback cycles		X					X
Processes focus on collaboration	X	X					X
Define releases regarding business objectives			X			X	
Self-service infrastructure provision and deployment (IAC)			X	X	X	X	
Optimization to customer KPIs			X			X	
Cross-silo analysis				X	X		
Quality gated commits					X	X	
Continuous deployment	X	X		X	X	X	X
Automated (continuous) build creation		X				X	
Continuous testing (automated recoverability)	X	X	X			X	
Automated documentation generation		X				X	
Feature toggles				X		X	

With that in mind, the model that is presented in the next section is based on the CMMI model, a fixed level maturity model. As described in the background, two representations of this model exist: continuous and staged. It has not become clear from literature that a standard sequence of improvements exists or that all capabilities need to be in place to proceed to the next maturity level therefore, the continuous variant of the fixed level maturity model is chosen as it provides more flexibility for the organizations.

In the existing literature, it was quite often chosen to come up with their own definitions for the DevOps maturity levels. However, because of how widely known and used CMMI is, to use the standard CMMI maturity levels seems to be the best option.

4.4 USING THE LITERATURE SOURCES TO PROPOSE THE FIRST VERSION OF OUR MATURITY MODEL V1

The previous sections provide enough material to create a first version of the DevOps maturity model. In this case, it was chosen to use the CMMI framework as a starting point, based on the arguments given in the discussion section. This framework will be combined with the capabilities found in the literature. The result of the first iteration of the DevOps maturity model is shown in Appendix C. The model is divided into the same sections as the dimensions of DevOps described in the background section: 'Culture', 'Automation', 'Communication & collaboration', 'Measurement' and 'Monitoring'. In the following sections, the model will be described for each level structured by describing each dimension for the specific maturity. Similarly, to CMMI, our proposed model includes five levels.

4.4.1 Initial

The initial level is where most organizations will be before they start implementing DevOps. This means that the teams are isolated and organized around one specific skill set. This leaves little place for feedback inside the organization. Many opportunities for process improvement exist as most processes are done manually.

No automation of the delivery process has taken place, environments are manually provisioned, builds are done by build scripts and deployment happens manually. Documentation and saved configurations of projects or customers are non-existent.

The isolated teams result in indirect communication between them, which mainly happens via management of the teams. This results in poor communication, as information get lost in translation between the various people involved. The isolated teams also influence the collaboration between the teams, as this is only on an ad-hoc basis

Reporting on the teams happens on an irregular basis and are made manually. These reports can be on various subjects, for example, on the stability of a recent build or an overview of test results. However, the reports are just visible to the report runner and are not available to the rest of the team.

The monitoring is done on an ad-hoc basis, as well as the testing. Quality standards are non-existent, and the quality is therefore only managed when there is a direct need. The quality of the code is monitored manually.

4.4.2 Managed

The managed level is where organizations are consciously implementing DevOps and the foundation to build on is set. The teams are stable, however still independent as they have one backlog per team. The teams are structured to focus on short-term deliveries. The processes and methods are defined for the team itself and rapid feedback can be given in a safe environment, but again only in the team itself.

The delivery processes are now scheduled and the environment in which they are released is partly automated and organized into modules (OS, Database etc.). Builds can be made and stored automatically, however, versioning is still added manually. The deployment is mostly standardized and even partly automated. The documentation and configurations are up-to-date.

The communication and coordination are facilitated and actively managed to provide an environment that motivates active collaboration in the teams.

Quality reports are now scheduled and created by the tools used in the process. The latest report is always accessible and visible to the whole team. The objectives for the software are linked to specific releases and the requirements are managed centrally.

A quality standard exists and is managed. This is achieved by doing systematic testing and specific requirement testing. After each build a test is performed as well and whenever the build is broken this is detected. The incidents resulted are gathered and prioritized together with the existing requirements.

4.4.3 Defined

This level provides a stable environment and already has quite a few of the capabilities that define DevOps implemented. The teams are structured around projects and processes are adapted to the collaboration between the teams. Feedback on the collaborations between teams can be given rapidly.

The delivery process is standardized and automated. Builds can be triggered by, for example, a code check-in by a developer. The versioning is done automatically and because the builds are standardized, they can be deployed on all environments. The environments are automatically provisioned, and virtualization is done where applicable, this process is standard across all the environments (test, staging, prod etc.). A basic deployment pipeline is created for all environments and, outside of production, it is possible to do the deployment automated, as well as most of the database changes that can occur with new build versions. Documentation and configurations are regularly validated.

The communication and coordination of teams are standardized and direct. The collaboration is further improvement by sharing pain between the teams. This creates common goals which further increase collaboration.

The reports are visible across the teams and are created strategically. The report history is available for comparison and a dashboard can be created to visualize this across the portfolio.

Advanced systematic testing is performed and integrated into the process. The management centralized and automatically, whenever a change to the code is committed. These changes are gated, which means they are committed into the main branch with version control, in case something breaks it is easier to revert. The results are compared to quality metrics to measure performance. The monitoring uses business and end-user context and the resources of environments to ensure stability.

4.4.4 Measured

This level is when most capabilities of DevOps are implemented. The teams are structured around the whole product and not only a delivery or a project, thus giving more responsibility. The teams are cross-functional and have a knowledge overlap on the knowledge area of the other team. The processes and methods are adapted throughout the software delivery process in the organization. Feedback is given frequent and on all processes regarding software delivery.

The complete automation of a frequent delivery process is finished. The environments are automatically managed, and the architecture is fully composed of components. Automated build creation can be improved by parallel processing on multiple build machines. Build metrics are gathered to improve the build automation and the build is also automatically deployed into production. The database changes are fully automated. Deployment causes zero downtime. Finally, the documents are updates based on the gathered experience and quality requirements.

The collaboration and communication are structured and in a constructive manner from peer-to-peer. Thus, no interference of management in communication between the people inside and outside the team. Collaboration is more intertwined with the processes as well.

Regarding the measurement, the focus is more on collaboration-based processes to identify bottlenecks and inefficacies. This is supported by a trend analysis of the reports. Automation metrics exist to assess the progress against the set business goals for further improvement.

Finally, the testing is extended by qualitative testing and advanced automated systematic testing. Services can be virtualized for testing purposes to minimize disturbance in other environments. The performance management is organized, and specific applications can be used for enterprise resolution. The code quality is automatically monitored at this phase.

4.4.5 Optimized

The final level is when all capabilities of DevOps are implemented and focuses on optimizing them. The teams are cross-functional, interdisciplinary and focused on KPIs. To support this the team members have become multi-skilled and flexible. The process is continuously evaluated and improved upon.

The automation is optimized to maximize throughput of releases. The environments are managed through IAC and all environment configurations are externalized and versioned. Builds are continuously created as well as the deployments to all environments (including production). The software contains feature toggles to minimize impact of new features and release those whenever needed. Documentation and configurations are automated generated at this level.

Communication and coordination are optimized according to the set metrics of the previous phase. This is further improved by community building to optimize the communication and collaboration.

Reports are created with an analysis of multiple departments (cross-silo). Releases are defined on specific business objectives and the KPIs are measured and optimized to increase customer value.

Testing happens continuously, as well as resilience testing. When errors occur, the software recovers automatically. The gated commits are now checked on quality before they are accepting into the software. This all contributed to continuous quality improvement of the software. Finally, the monitoring is optimized regarding the customers' KPIs.

4.5 HIGH-LEVEL REPRESENTATION OF THE DEVOPS MATURITY MODEL V1

The maturity model described in the previous section is detailed, this can help companies get a better understanding of their DevOps maturity, but also is the size of two pages. To improve the understanding of the maturity model it is helpful to visualize it on a more high-level overview. This can be used to explain how the model is build up and how to interpret it. This results in the model of figure 4.4

This model is shaped as a house, with DevOps being the roof, because the capabilities are the fundament for a thriving DevOps environment and thus, metaphorically, holding up the roof. The overarching connection between the capabilities and the DevOps environment in this model, are the maturity levels in this model, and thus in-between the roof and the rest of the house. In the literature review it has become clear that culture is an important fundament for supporting better communication & collaboration. But it also supports the measurement category, as reporting and requirement management must be supported by the process improvement and feedback cycles stemming from the culture category. The monitoring and automation are stand-alone as implementing these does not necessarily need any of the other categories, however all categories are needed to provide the best DevOps environment.

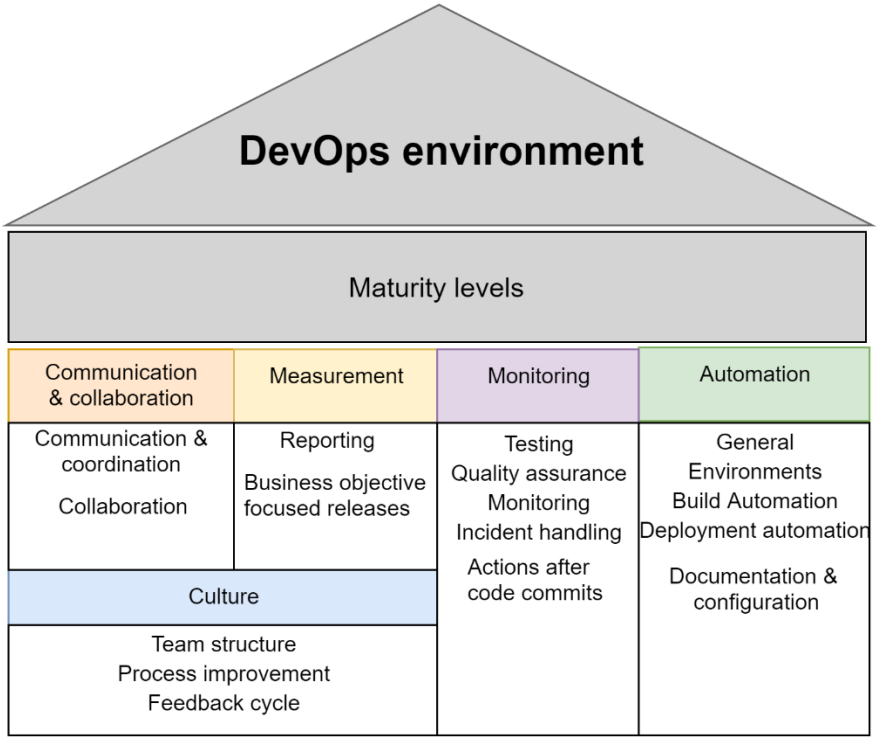

Figure 4.5: High-level maturity model V1: an overview

5 EXPERT INTERVIEWS

The goal of the DevOps maturity model is to be applicable in practical situations, therefore it is important to interview experts in the DevOps field on which capabilities they think that are necessary for successful implementation of DevOps in organization. This will be the first round in the expert interviews and will result in the second version of the DevOps maturity model

The DevOps maturity model and its capabilities must then be validated to ensure the credibility of the model. This is done by means of expert opinion in which the model will be subjected to experts in the DevOps field. They are asked to imagine how this model will interact with problem contexts imagined by them and then predict what effects they think this would have [51]. If these results do not reflect the intended use of the model than it must be redesigned. This chapter will outline the process that was performed and the results that followed from it.

5.1 INTERVIEW SET-UP

The interviews taken with the experts were all semi-structured [52], this means that the interviewer adhered to an interview protocol but could divert if something came up that had to be explored further. The interviews were done in two rounds. The first round comprised of questions about their experience with DevOps and associated capabilities that were needed in different DevOps stages. This data was used to create the DevOps maturity model v2. The second round, performed on a later date, was an open discussion on the DevOps maturity model v2 for validation purposes.

These interviews were held on-site at three different organizations with a total of five people with different backgrounds. The first three interviewees were people from the same company, but in three different roles, all with practical experience in DevOps. This company employs more than 1000 people and is an outsourcing company in Vietnam. The fourth interviewee has both practical and theoretical background, with more than 10 years of work experience in information systems and a PhD. in information system management. The third organization employs about 600 people and is also an outsourcing company in Vietnam. A summary of the interviewees can be found in table 5.1.

When the interview was finished, the most important parts of the interview would be transcribed, if the organization policy of the interviewee allowed recording, and summarized. As the interviewee's struggled with English, it was necessary to rewrite sentences said by the participants for the purpose of readability. However, the interviewer tried to not change anything to the essence of what was said. The results and quotes used in this report, from the interviews were sent back to the interviewees, who validated the results and made sure that the transcription did not change the meaning of the sentences.

Table 5.1: Overview of interviewees

Name (acronym)	Organization	Role	Experience in current role (years)
Interviewee A (Iv-A)	Organization A	DevOps team manager	5
Interviewee B (Iv-B)	Organization A	DevOps architect	10
Interviewee C (Iv-C)	Organization A	DevOps engineer	3
Interviewee D (Iv-D)	Organization B	University lecturer	6
Interviewee E (Iv-E)	Organization C	DevOps engineer	4

5.2 DATA ACQUISITION AND ANALYSIS – EXPERT INTERVIEW

The interviews were taken to gather data on which capabilities are important for successful implementation of a DevOps environment inside an organization. The interview was started by gathering some general data, like the background and years of experience of the interviewee. The rest of the structure is comparable to the dimensions of DevOps described in chapter 2. It was chosen to start with the dimension of automation as this is a hard-skill which is well known in Vietnam. The rest of the conversation was split between the other dimensions. The full protocol followed for the interviews can be found in Appendix D.

5.2.1 Analysis of interviews

Some of the interviews were recorded, however not all interviewees provided permission to record the interview. For these interviews it was crucial to maintain good notes during the interview and process the interviews right after they were held. The recorded interviews were manually transcribed, however the introduction, ending and some in-between conversations were omitted, the rest of the transcript was verbatim. The interviews lasted between forty and sixty minutes per interviewee.

The interviews were analyzed by applying coding practices described by Saldana [53]. According to his book, the analysis can be split up into multiple coding cycles. For the first cycle, the 'initial coding' method, also known as 'open coding', was chosen that belongs to the sub-category of elemental methods. Elemental methods are described as the foundation for future coding cycles. Specifically, 'Initial Coding' is described by Saldana as the "first major stage of a grounded theory approach to the data" [53]. This description makes it clear that there is

a need for a second coding strategy to continue the foundation of the 'Initial Coding'. This is where the second coding cycle comes into play, this cycle is described by Saldana [53] as finding the "bigger-picture" and as the latter stages toward developing a grounded theory. In this cycle it is chosen to use 'Axial Coding', which is a method to develop broader conceptual categories based on the identified relationships between the codes created during the initial coding. These two coding cycles in the context of the interviews is graphically presented in figure 5.1.

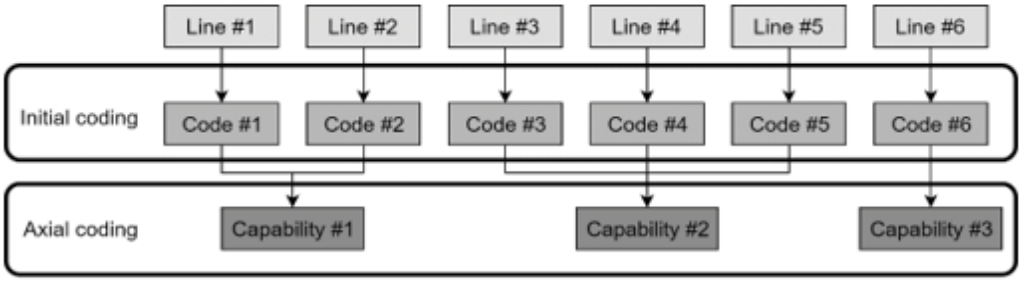

Figure 5.1: The coding process used in this thesis

In the process of doing the initial coding, only the responses of the interviewees were considered and not the interviewer questions or comments. In those responses it was important to search for processes, which are the participants actions or words. For example: "*I think these days it is important to teach students about collaboration in DevOps, because it is important that teams can work together interchangeably and know what is necessary to help each other function better.*" The codes given to this sentence are, 'collaboration', 'team collaboration', 'work interchangeably' and 'improve others'. This example and another one is given in figure 5.2.

Interview answer #1

"I think these days it is important to teach students about ***collaboration in DevOps****, because it is*
Collaboration

important that ***teams can work together interchangeably*** *and know what is necessary to* ***help each***
Team Collaboration; work interchangeably Improve others

other function better

Interview answer #2

DevOps is all about ***Automating the builds*** *and* ***testing these builds automatically.*** *Testing can be on*
Build automation Automated build tests

bugs, code coverage, unit testing*. Trying to accomplish this will result in* ***less mistakes*** *and*
Thorough testing Stable products

happy customers
Customer requirements

Figure 5.2: Initial coding

This process of 'Initial Coding' was done throughout the interviews and based on the transcripts and notes. Each sentence or paragraph got one or multiple codes. Most codes were related to the capabilities from the literature review because of

the use of deductive coding described by Miles and Huberman [54]. However, the coding was not restricted to the already discovered capabilities, because the goal is also to discover new capabilities.

After this was finished, the second cycle of coding was started: 'Axial Coding'. During this process the codes were sorted and re-arranged to help the process of creating broader conceptual categories that cover multiple codes as was described in figure 5.1. An example is the combination of the categories 'thorough testing', 'automated build tests' and 'stable products' discovered during Initial coding, into the capability 'Testing'. This example, and others that followed from the coding in figure 5.2, are shown in figure 5.3

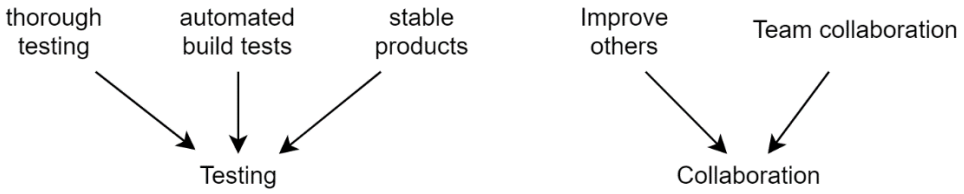

Figure 5.3: Axial coding

5.2.2 Results

The results will be described according to the structure that was used in the interview, starting with automation, followed by collaboration & communication, culture, measurement and finally monitoring.

Automation
The first capability that was constructed from the interview data was focused on a fully automated DevOps cycle. All the participants mentioned that DevOps is to be considered to not only be a fully automated build process, but also automated testing and automated deployment. Iv-A explained this as:

"*The scrum methods in development was focused on continuous development, but DevOps is focused on combining continuous integration and continuous delivery*" .

Iv-A continued that this does not only include the software product but also environment envisioning. Something that was currently invested in is the use of infrastructure as a code, by using various tools which can result in getting an environment ready with one click.

The second capability regarding DevOps came from Iv-C, who is busy managing DevOps tooling every day. Logically, Iv-C considered tooling to be very important to a successful DevOps process. Iv-B agreed on this, but also mentioned that tools are restricted to open-source because of limited funding for projects. Fortunately, many good open source tooling exists for DevOps. The one mentioned mostly was Jenkins for managing the continuous integration.

The final capability regarding automation is the creation of microservices. Creating an application from separately deployable services that perform specific functions. Instead of monolith software, where the whole product is one application. Iv-C stressed that:

"*The life of both operations and development becomes much easier with microservices because if something is unstable than the rest of the application can continue to work.*"

Collaboration & Communication
The interviewees were mainly focused on collaboration between teams. They described this as:

Iv-D: "*Shorter development cycles require closer collaboration between the different teams, which is something we teach at the university. We also circulate the roles for the students, so that they will get familiar with this way of working*" .
Iv-E: "*The members of each team that takes part in DevOps will take a part in another team every now and then, this creates more understanding between the people*"

This translates to the capability of having interdisciplinary employees that will be able to change roles and therefore better understand the needs of others.

Iv-B explains that collaboration is something they try to improve, but that this is harder to achieve because people can be "lazy" and just focused on their own work. Iv-C thinks that collaboration can be improved by sharing targets and goals, this will create less individualism and therefore more teamwork.

Iv-A describes communication for DevOps as:

"*Direct communication is very important, the cycles in DevOps are so short, that communication between teams cannot lack behind*" .

Resulting in the capability of direct communication and not having to go through the hierarchical layers.

Culture
The culture is not the first capability that came to mind when asking interviewees to describe DevOps. However, when asking specific questions about culture it was deemed very important by all interviewees, however, also to be the most difficult capability to achieve. Also, Iv-A and Iv-C had difficulty describing how the right culture could be established and what exactly the ideal culture for DevOps implementation consists of.

Iv-B presented a print-out of a DevOps lifecycle model, that is shown in figure 5.4, to explain the need for a culture to foster a DevOps environment. Iv-B described it as:

"*DevOps is the full automated cycle, and a subcategory of Agile methodology, as DevOps is all about moving faster and more agile processes. This makes it important to create an agile mindset throughout the DevOps lifecycle, and even throughout the organization. However, it seems to be hard to convince people from other teams and management of this need, so focusing on the operation and development teams is enough for now. To improve further it is important to have the leadership committed to it*" .

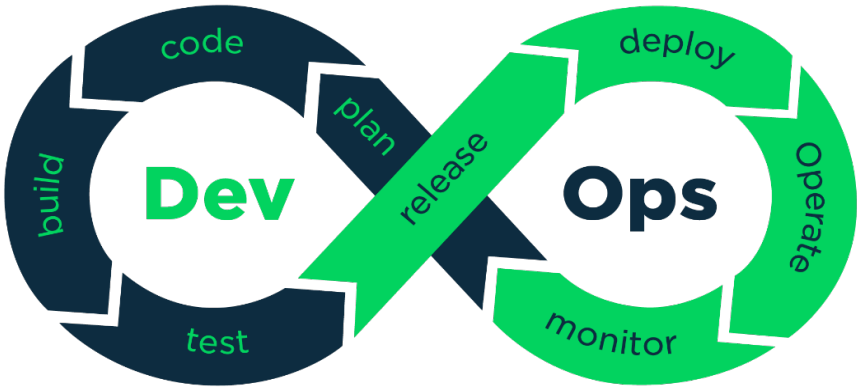

Figure 5.4: DevOps cycle [3]

The capability that can be taken from the interviews is that the culture in teams and the organization should be very agile to foster a successful DevOps environment.

Measurement
When the interviewees were asked questions about measurement in a DevOps environment, the main matter that was talked about is the reporting on automatic builds and tests. The tooling that is used in organization A can provide automatic reports on successful builds, code coverage of tests and successful unit testing among other things, as reported by Iv-A, Iv-B, Iv-C. According to Iv-B:

"*The reports generated by the tools are used a lot and also other reports that are not automatically generated. These reports are enough in most situations, however, sometimes customers ask for more specific reports and we will create those. I think we can do much more, by trying to see trends for certain products or environments*."

Iv-D explains that reporting is not something that is being taught at the university, due to time constraints and because it is not deemed as important as the other software developing aspects.

Iv-A states regarding reporting in DevOps that it is important to have reporting available, but that not everybody can interpret them correctly. For example, code coverage testing will not need to be 100% and is almost impossible to achieve on big applications. This should be used only as a rough number. A capability that is needed in this case would be to have trainers and/or facilitators.

Monitoring
The first response to monitoring by Iv-A is:

"*Monitoring is very important throughout the whole process if there is a code conflict it should not be merged into the main code, if a build is failed an appropriate response has to be made, if the physical capabilities of the servers are overloaded something has to be done. Shortly, with monitoring a lot of melt-downs and other incidents can be prevented*."

This was confirmed by the interviews with Iv-B, Iv-C, and Iv-D. These result in the capability of quality gated commits and continuous monitoring.

The second capability that became clear was again voiced by all the interviewees, is the need for continuous testing. Iv-D states:

"*Testing used to be the not sexy thing of programming, so my student would be very reluctant to test the code they wrote themselves and most of the time we do not have enough time to force them to do it. However, testing becomes a more important part of the curriculum, because the more complex systems get, the more important this will be.*"

Iv-A confirms this and states that continuous monitoring is not without continuous testing regarding DevOps.

All the findings that emerged during the interviews are summarized in table 5.2

Table 5.2: Capabilities emerging from interviews

Capability	**Interviewee(s)**
Fully automated DevOps cycle (build creation, testing, deployment, environment provisioning)	Iv-A, Iv-B, Iv-C, Iv-D
Need for open source tooling	Iv-B, Iv-C
Microservices instead of monolith structure	Iv-C
Reporting trend analysis	Iv-B
Report visibility	Iv-A
Cross-functional teams	Iv-D, Iv-E
Having shared KPIs and knowledge overlap	Iv-B, Iv-C
Direct communication	Iv-A
Leadership commitment	Iv-B
Quality gated commits	Iv-A, Iv-C
continuous monitoring	Iv-A, Iv-B, Iv-C, Iv-D

5.2.3 Discussion

The first revelation during the interviews is that the soft-aspects of DevOps do not get the same amount of attention as automation. The interviewees knew a lot to talk about automation and the tools being used, but when the conversation steered to the soft-aspects the information became very limited. However,

when looking at literature, especially soft-aspects of DevOps are deemed important.

When questioning the interviewees about this, it became clear that the reason for the lacking information on the soft-aspects of DevOps is that not much is invested into soft-skills and that it is hard to convince the leadership to invest more. This can create friction in fostering a full-fledged DevOps environment.

Another interesting observation was the need for DevOps tooling. This capability is confirmed in the white-paper by Bahrs [45] and therefore added to the maturity model. An interesting nuance given during the interviews is the need for open-source tooling, instead of closed-source. The reason that was given, is that the cost in outsourcing projects have a major impact on the overall profitability. This nuance is not added to the capability in the DevOps maturity model, as this seems to apply specifically to outsourced projects and the goal is that the model can be applied to any project.

What surprised us during the interviews is that all the interviewees mentioned, that Asia is always lacking behind Western countries in adapting to new methodologies. Unfortunately, they were not able to give examples for why this is their opinion. Therefore, we will provide some speculative reasons. For example, if they lack behind in DevOps adoption then it could be due to the possible differences in culture needed for implementation in (South)East-Asia. Or the DevOps implementation is different in (South)East-Asia, but they judge it through the Western idea of DevOps. However, during the interviews it did not actually seem that they lack behind. They could have many reasons for saying that they lack behind, while this is not correct. For example, the interviewees look up against Western society and think they are better in general. Or to seem humble towards the interviewer. The exact reason is unknown; however, the final version of the maturity model could be used to compare the Western and (South)East-Asian DevOps implementations to at least determine if they are lacking behind.

The interviews have not brought many additions or changes to the model, a reason could be that the DevOps philosophy is taught the same in these organizations as it is elsewhere. Nevertheless, many findings from literature have been confirmed, which provides additional assurance that these capabilities belong in the DevOps maturity model.

The findings presented in our results are added to the table that compared the literature on its capabilities. This gives a complete overview of all the mentioned capabilities in the literature and the interviews. This table can be found in Appendix B. This overview confirms that none of the points mentioned in the interview had not already been mentioned in existing literature. However, some capabilities that were discarded, because they were only mentioned once in the literature have been mentioned in the interviews as well. Therefore, it is decided to add these capabilities to the maturity model. The capabilities are regarding DevOps tooling, monolith to microservices architecture and continuous monitoring. This results in the DevOps maturity model v2, which can be viewed in Appendix C.

These results also warrant changes to the high-level overview of the DevOps maturity model, which was first presented in figure 4.4. However, we note that from high-level perspective, the changes are minimal, as most changes took place in the details of the model and not in the overall capabilities or categories. The only change was the addition of DevOps tools as a capability to the Automation category as can be seen in figure 5.5 (see the 5th item in the box below Automation).

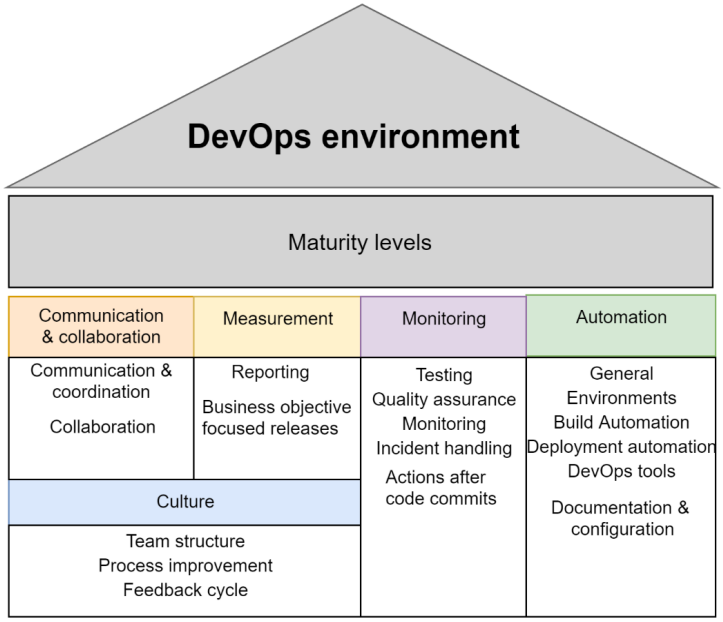

Figure 5.5: High-level maturity model V2: an overview

5.3 VALIDATION ROUND– EXPERT OPINION

The DevOps maturity model V2, created from literature and expert interviews must be validated. Our first validation was carried out as an evaluation study in which we collected experts' feedback on our model proposal V2. The session with each expert started with an explanation of the model and the questions focus on evaluation of the understandability, completeness and relevance. These aspects are inspired by the information quality measures of DeLone and McLean [55] for e-commerce and the evaluation criteria of March and Smith [56], but adapted to what we think is important for the DevOps maturity model. The evaluation was qualitative in nature. After each interview the model was updated, and the improved model would be presented to the next interviewee, this process is shown in figure 5.2.

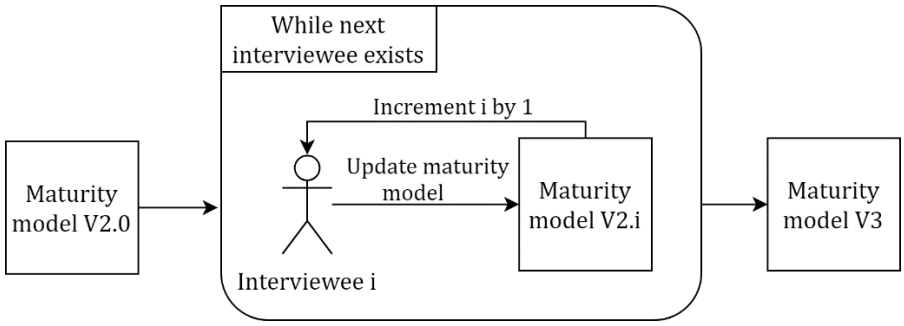

Figure 5.6: Maturity model intermediate updating process

The interviewees are the Iv-A, Iv-B, Iv-C and Iv-D that were already interviewed for data acquisition, Iv-E was not available for the validation round. The same acronyms will be used for the interviewees in the next section.

5.2.1 Results

In this section the results of the first validation will be presented by highlighting the points made by the interviewees during the interview. These will be presented 'as is' and will only be discussed in the discussion section.

Iv-A was surprised by the size of the model and questioned the readability of the model:

"*Usually these models fit on one page, which creates an overview that can be understood in a short amount of time. It may be hard to create similar readability of a model that covers two pages*."

However, after some time was given to study the model more in-depth the interviewee did get a full understanding of the model. It was mentioned that especially the categories in the first column (the DevOps dimensions) helped with the understanding.

The categories were discussed in-depth to validate that the capabilities were in the correct categories. Iv-B recommended to make some changes to the categories:

"*The feedback cycle is not in the correct category in my opinion, as this is directly related to communication it should be in the communication & collaboration category. Also, I think that communication & collaboration are too closely related to culture to be in its own category.*"

Iv-A did not comment on the same categories but thought that automatic testing should be part of the automation category and not the monitoring category. The other interviewees did not bring up issues with the categories.

Iv-A, Iv-B, and Iv-D thought that the levels were confusing and should be changed.

Iv-A: "*Level 5 is the optimized level, but is this really necessary to go this far? Do the benefits still outweigh the costs? I do not think that we would ever go this far in our company.*"
Iv-D: "*I think you should rename the levels to make it clearer what the difference is. Not everything in level 5 is focused on optimizing for example. Otherwise, it is necessary to reorder all the capabilities to make them more fitting to the current names of the levels.*"

During the interviews each expert was asked to have a close look at all the current capabilities in the DevOps maturity model and to note if any are missing, to improve the completeness of the model. Iv-A and Iv-C did not notice any missing capabilities; however, Iv-B and Iv-D mentioned the following regarding the capabilities:

Iv-B: "*The capabilities cover the monitoring of the systems, code etc. But what happens when something goes wrong? How are incidents handled?*"
Iv-C: "*This DevOps maturity model is detailed, but it does not become clear how to achieve the capabilities, would it not be helpful if example steps were given to support the implementation of these capabilities?*"

Finally, according to the experts, clarification was needed for some capabilities.

Iv-C: "*I think some capabilities are too difficult to understand, maybe change the names, or make it clearer in another way.*"
Iv-D: "*I had to ask you for some terms what they mean, and I know much about this field of research, so maybe change some of the terms*"

The overall conclusion of all the interviewees at the end of the validation was that the model could be helpful for the implementation and assessment of DevOps. Iv-A especially emphasized the possibility of using the model to increase awareness in the areas of DevOps that are otherwise neglected. Iv-B mentioned that the model seems very complete in covering all the DevOps areas.

5.2.2 Discussion

Our first validation exercise provided some interesting changes to the maturity model. These changes will be discussed in this section and finally be presented in the 3rd version of the maturity model.

The first suggestion was the change to **the naming of the five levels**, which is thought to be improving the understandability of the model. This is an interesting decision because after the literature review it was explicitly chosen to follow the CMMI levels, because of how widely known this model is. Also, the naming of maturity level also did not come up during the earlier interviews. However, the naming was brought to our attention again by the first interviewee in the validation round. This change to the names was also discussed with the other interviewees and each of them agreed that these new names create more clarity. As described in the literature review, different names are also used in the white papers of IBM and InfoQ [45]–[47]. It is important to not only contribute to the scientific literature but to also create a model to be used in practice. Therefore, it is decided to change the names of the levels for the third version of the DevOps maturity mode. The new levels are **'base'**, '**beginner'**, '**intermediate'**, **'advanced'**, **'expert/extreme'**. The final level is chosen to be 'expert/extreme' because as described before, this level is not suitable for many projects. However, for some projects, it can be very useful and would warrant an expert level.

Another suggestion made by one of the interviewees is to make the model more compact to increase the **readability**. However, the goal of the model is to be fine-grained enough to provide a comprehensive guideline in adopting a DevOps environment. Therefore, it is chosen to keep the size of the model the same as the second version of the DevOps maturity model.

It was also mentioned to add a capability for **incident handling**. Currently the model does cover automatic recovery of systems, however, we agree that it is not specific enough on the handling of incidents in general. A capability is added to the maturity model category monitoring. It was also mentioned to elaborate on the application of the capabilities, however, another interviewee already mentioned that the model was lengthy. Therefore, it is chosen to not make it even longer by explaining how to implement the capabilities in the model as was suggested by another expert.

The next suggestion that was made to change some of the orders that the capabilities were in in the maturity model. As well as to which capabilities belong to which DevOps dimension. However, this categorization of the capabilities followed from the existing literature. Also, the experts did not share the opinion on what category should be changed or which capability should be shifted. Therefore, it was chosen to remain the same categories and the same capabilities linked to them.

Finally, some confusion existed on **the names of the capabilities**. Therefore, following the interviews, each capability was taken a critical eye at and improved on clarity where necessary. This resulted in some small trivial changes being made,

that did not influence the model greatly except for its clarity and are therefore not independently discussed in this section.

In conclusion, the validation of the experts did not result in many changes, but it was helpful nevertheless as changes that were made, increased the understandability. A reason for the limited amount of changes could be that these same experts also contributed to the model in the first round of validation. The changes that were made can be viewed in the DevOps maturity model V3, presented in Appendix C.

The changes made to the DevOps maturity model V3, also result in the final iteration for our High-level maturity model. Only two changes to this model were made: First, the addition of the capability 'Incident handling' to the category 'Monitoring'. Secondly, the change in the name of the capability 'Business objective focused releases' to 'Requirement management'. These changes can be seen in figure 5.7.

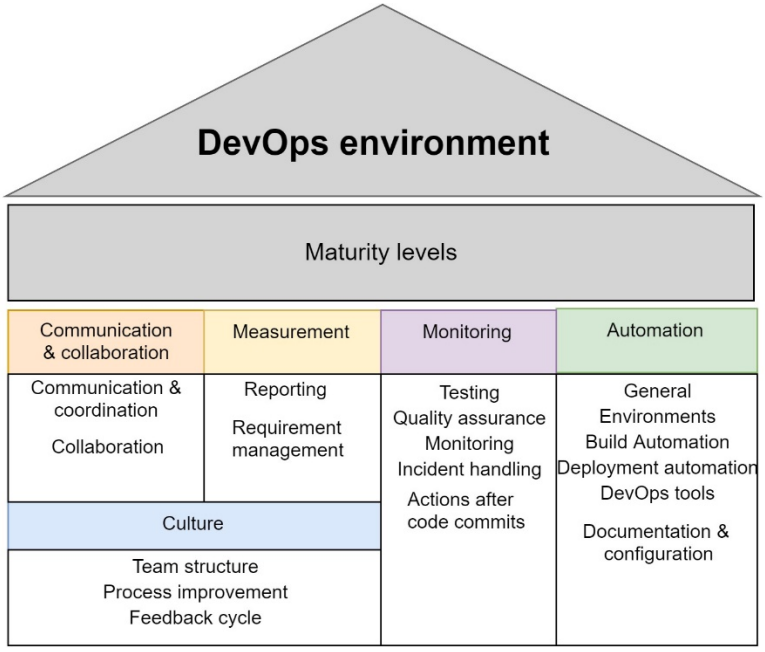

Figure 5.7: High-level maturity model V3 overview

6

CASE STUDIES

The validation of the previous section resulted in the final iteration of the DevOps maturity model (V3). The next step is to evaluate the model in practice. Before the case study started, a case study protocol was created and the plan validity was tested according to the guidelines as recommended by Yin [57]. The results of this plan validity can be viewed in Appendix E.

The case study was performed by sending out assessments to people in various roles and in four different organizations. In one organization this was combined with observations and informal conversations to create an even more comprehensive image. The other organizations were found either through connections of the first organization or via a posting made on February 20, 2019 in a Facebook group named 'DevOps Vietnam' in which people with multiple years of experience in working in DevOps were asked to complete the assessment. This posting was reposted once, a week later to gather more people. After two weeks, enough people had given a response for me to start processing the results. Each result was verified to be coming from a person working at an IT organization, the results that could not be verified were disregarded. A complete overview of each case is provided in table 6.1.

Table 6.1: Overview of cases

Case ID	Organization	Years of running the project	Number of people employed	Number of teams involved	Characteristics of software developed
Case 1	A	2	30	3	Internal project
Case 2	A	3	40	5	Python based cloud software
Case 3	A	6	80	7	Locally hosted software
Case 4	A	1.5	20	3	Cloud based and hosted in AWS
Case 5	D	8	20	3	SAAS application
Case 6	E	4	60	4	Outsourcing project
Case 7	F	4	120	6	Cloud based
Case 8	F	Less than a year (8 months)	15	2	Internal pilot project

The first four cases are on organization A, which was described in chapter five, but it is useful in this context to add that DevOps in this company has been applied to many of their projects. It is continued to be invested in for improvement of roll-outs and updates. This applies to both the customer sites and internal projects.

The fifth case was done at organization D. This organization started in 2008 in Vietnam and employs about 170 people and creates end-to-end software solutions. The focus of this organization is on creating outsource solutions.

Organization E was used for an assessment in the sixth case. This organization provides offshore product development, software outsourcing, and independent software testing services operating in Vietnam. The organization is listed in the top 100 of best working places in Vietnam. The type of software developed in the project that was assessed is not disclosed, besides it being an outsource project, because of reasons given on proprietary grounds.

The seventh and eighth cases were the assessments of projects at organization F, established in 2004 as an internet and technology organization in Vietnam. The organization is ranked in the top 100 best IT companies in Vietnam.

The organizations were asked about the benefits that they achieved by implementing DevOps, however not every interviewee could share that due to the organizations' proprietary information policy, which is shown in the table by (-).

Table 6.2: Overview of achieved DevOps benefits for each case

Case ID	Organization	Benefits of implementing DevOps
Case 1	A	Decreased time to production & insight through tooling
Case 2	A	Decreased time to production, smoother processes, preventing incidents, far advanced automation
Case 3	A	Build automation, more collaboration between teams
Case 4	A	-
Case 5	D	Automation of the product cycle, reducing bugs, reducing incidents at operations and aligning the customer requirements with the product.
Case 6	E	Automated steps in the product lifecycle
Case 7	F	-
Case 8	F	-

6.1 CASE STUDY RESULTS

As intellectual property is considered very important in Vietnam, the participants required full anonymity for their participation and the projects are therefore only shortly described on some of the project characteristic. The assessment of the project results in a completed DevOps maturity model, with selected maturity levels for each capability. The level was only selected if all requirements were fulfilled. If only part of the requirements for a certain capability level was fulfilled, a lower level would be selected. These results will be summarized and presented for each case. Below, we present our results as related to each case. We report some details about the evaluation on the success of the DevOps maturity model and some recommendations for future changes to the model.

6.1.1 Case 1

The first case study is on an internal project that is taking place at organization A. This project involves about 30 people and has been running for 2 years. The teams involved are development, quality assurance and the DevOps team (the DevOps team takes care of the operations side). The software involved is hosted internally. The assessment was completed by a DevOps engineer. The benefits achieved by the DevOps implementation, as described by the DevOps engineer, was insight into new software builds and reduced lead time to production. The results of the DevOps maturity assessment are summarized in figure 6.1.

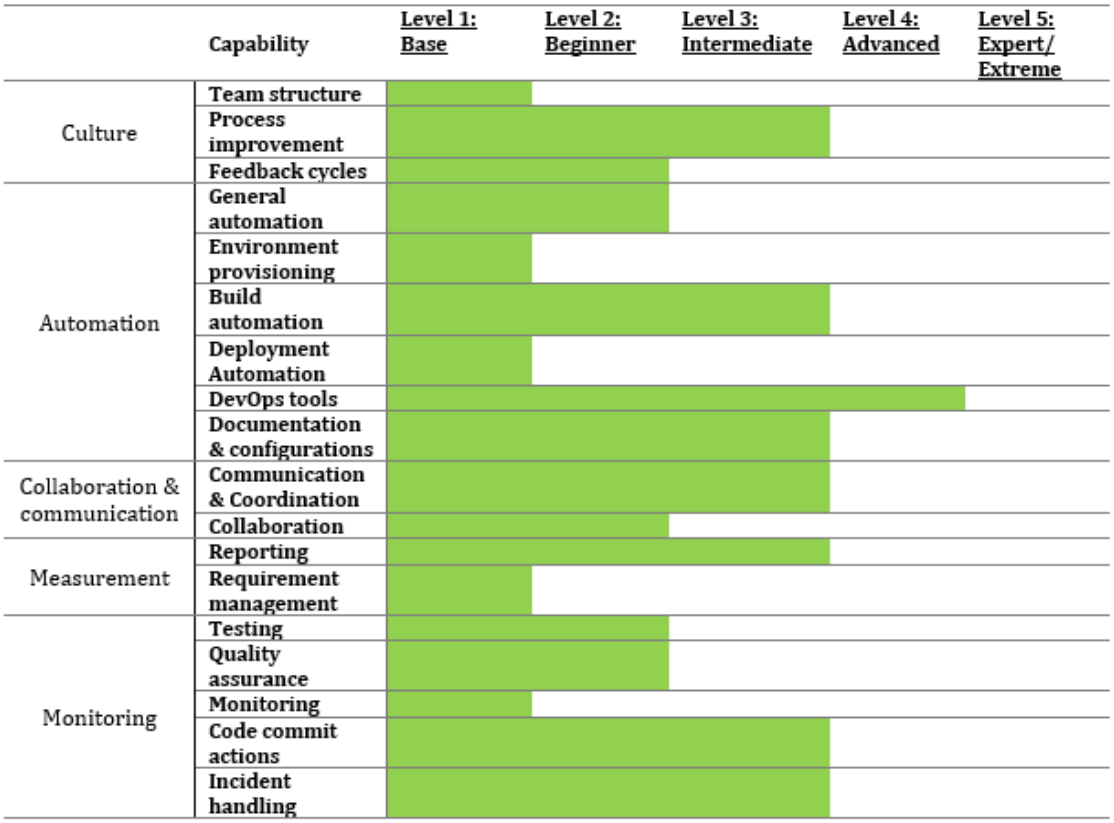

Figure 6.1: DevOps maturity assessment results case 1

The first aspect that becomes clear when looking at the results from the assessment, is that the maturity levels are quite dispersed, from level 1 to level 4. This is the case for all five categories of capabilities. This indicates that it is quite easy to life the DevOps environment to the next level if the team structure, environment provisioning, deployment automation, requirement management and monitoring are improved. However, it is questionable if this is necessary as the project is internal and therefore will be held to different standards by organization A.

6.1.2 Case 2

This case study also took place at organization A. The project runs for about 3 years continuously and with more than 20 people involved spread across three different teams. The software product is a python cloud base project hosted on AWS. The perceived benefits from DevOps are reduction in time to production, smoother processes, less incidents, preventing incidents from happening and far advanced automation. The results of the DevOps maturity assessment can be seen in figure 6.2.

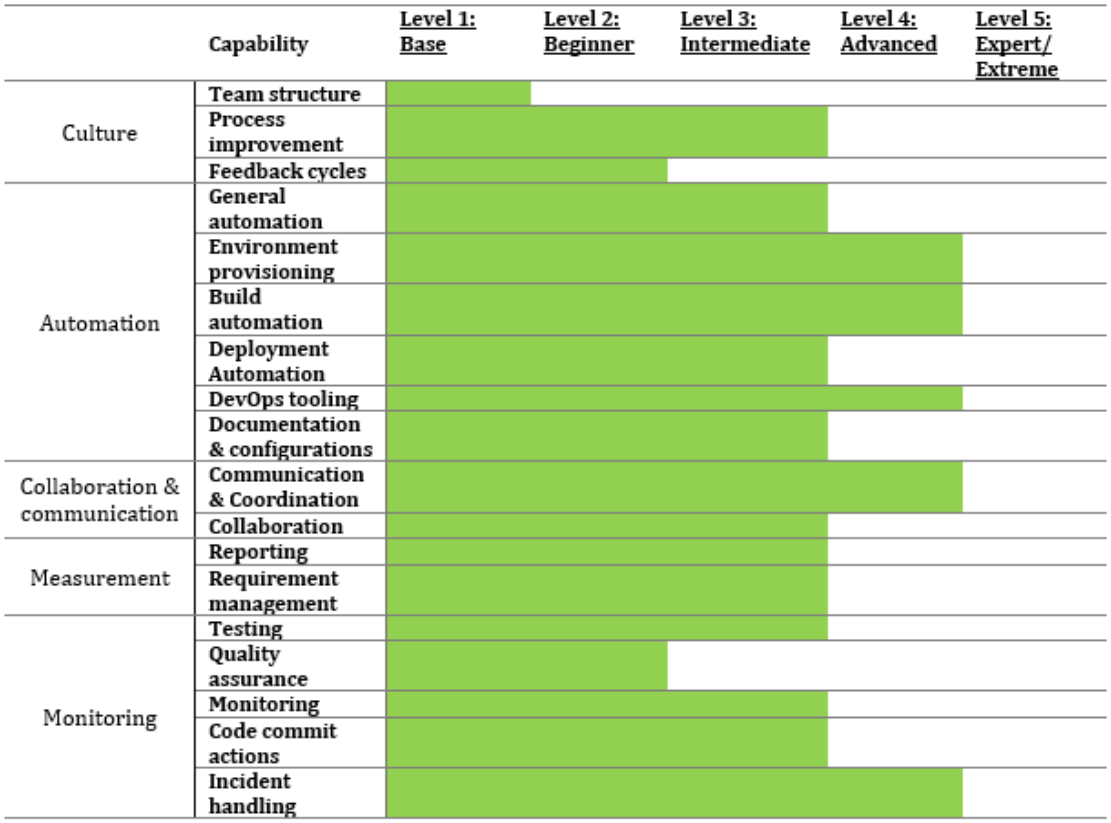

Figure 6.2: DevOps maturity assessment results case 2

Analyzing figure 6.2 gives quite a low variance in the maturity levels. Only the team structure is on level 1, however this is also one of the hardest to change as the organization wants to have the people divided based on their skillset. All the other capabilities are on at least level 2, but mostly on level 3. This indicates an intermediate implementation which could be suitable for most projects.

6.1.3 Case 3

The project for this case study involves a project that was started 6 years ago with a waterfall development approach. This approach has changed during the past 3 years to an agile approach. The projects have about 80 people involved spread across 7 teams, including multiple software development teams. The software is hosted internally for testing purposes and locally at the customer in staging and production environments. The perceived benefits of implementing DevOps were build automation, closer collaboration between the teams involved. The results of the DevOps maturity assessment can be seen in figure 6.3.

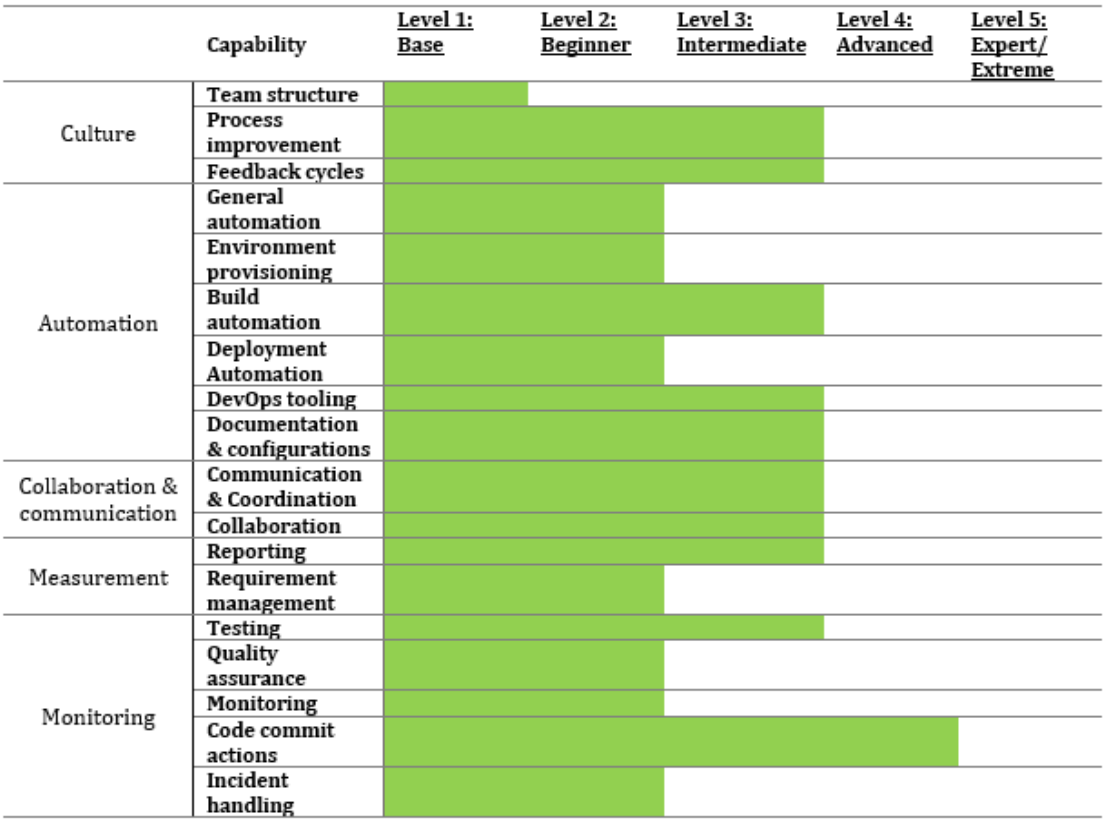

Figure 6.3: DevOps maturity assessment results case 3

Looking at figure 6.3 gives the indication of a stable maturity level between level 2 and 3. Again, only the team structure is level 1 as is by the choice of organization A. The actions performed after a code commit are level 4, which could be very helpful in keeping the quality up throughout the lifecycle, as coding issues are already taken care of before merging it into the main branch.

6.1.4 Case 4

The project that was assessed for case 4 is as far as organization A has taken DevOps in any project. The reason that this project was chosen was because of a high rate of releases being developed and relatively recent started only 1.5 years ago with an agile mindset. The project has about 40 people involved and consists of 5 teams. The product is cloud-based and hosted in AWS. The results of the maturity assessment can be seen in figure 6.4

Figure 6.4: DevOps maturity assessment results case 4

The project for case 4 seems quite mature as most capabilities have reached the level of intermediate. Especially the automation part has been taken good care of, as many have already reached level 4. The team structure is low by choice of the organization. The next capability with the lowest level is quality assurance at level 2. If quality metrics are created to measure performance improvement, then the whole project can be considered level 3.

6.1.5 Case 5

This is the first project that was assessed that is not part of organization A. This project takes place in organization D. This project consists of about 20 people involved, spread over teams of development, operations and QA. The project is hosted on a cloud service and provided to customers as a SAAS application. The project has been running for about 8 years. The perceived benefits by implementing DevOps in this project were automation of the product cycle, reducing bugs, reducing incidents at operations and aligning the customer requirements with the product. The results of the assessment are shown in figure 6.5

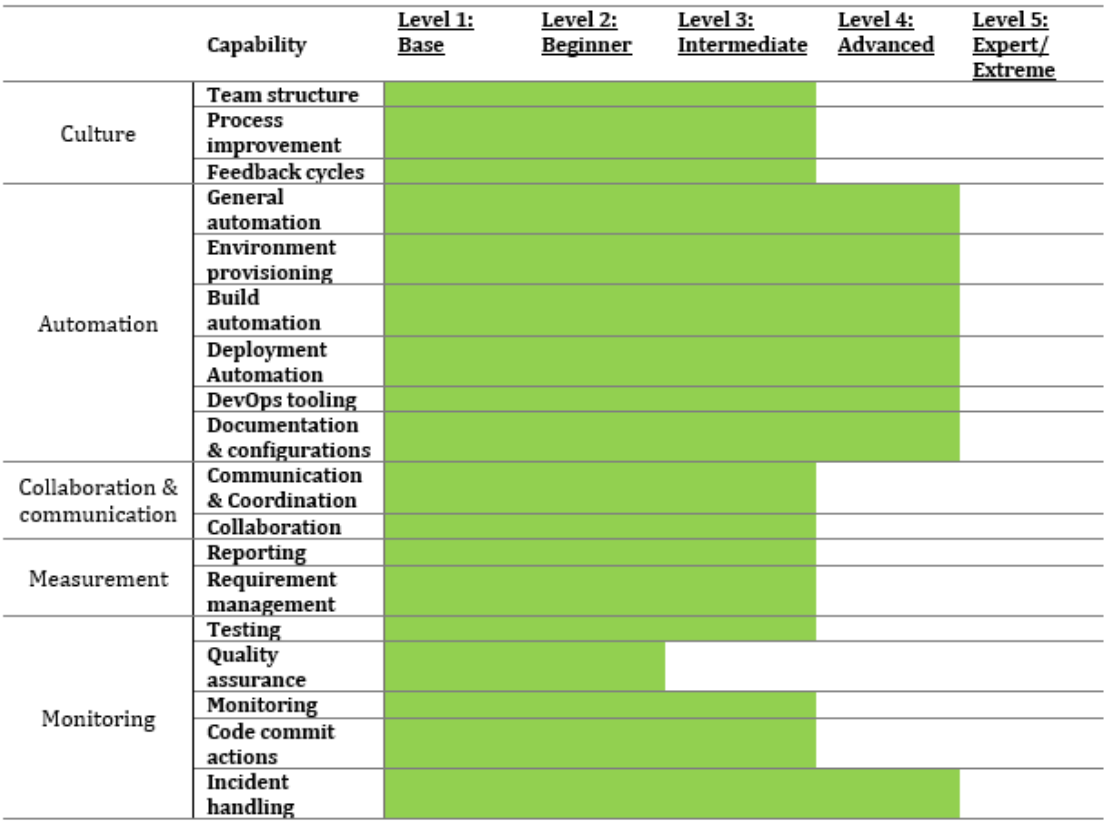

Figure 6.5: DevOps maturity assessment results case 5

Figure 6.5 provides a clear image. This project is almost completely at level 3, with the only exception being the quality assurance. Which can be taken to level 3 if the performance improvement metrics are created and used. Automation is fully brought to level 4, so again it becomes clear that this is the highest focus for this project, nevertheless the soft-aspects are impressive as well and probably at a level suitable for most projects.

6.1.6 Case 6

The project at organization E is an outsourcing project developed in-house for a customer abroad. The project was started 4 years ago with about 60 people involved, spread over four teams and is focused on creating a lighter workload by using the automation in a DevOps environment. The perceived benefits relating to the DevOps implementation at this project are automated steps in the product development cycle. The results of the assessment are shown in figure 6.6

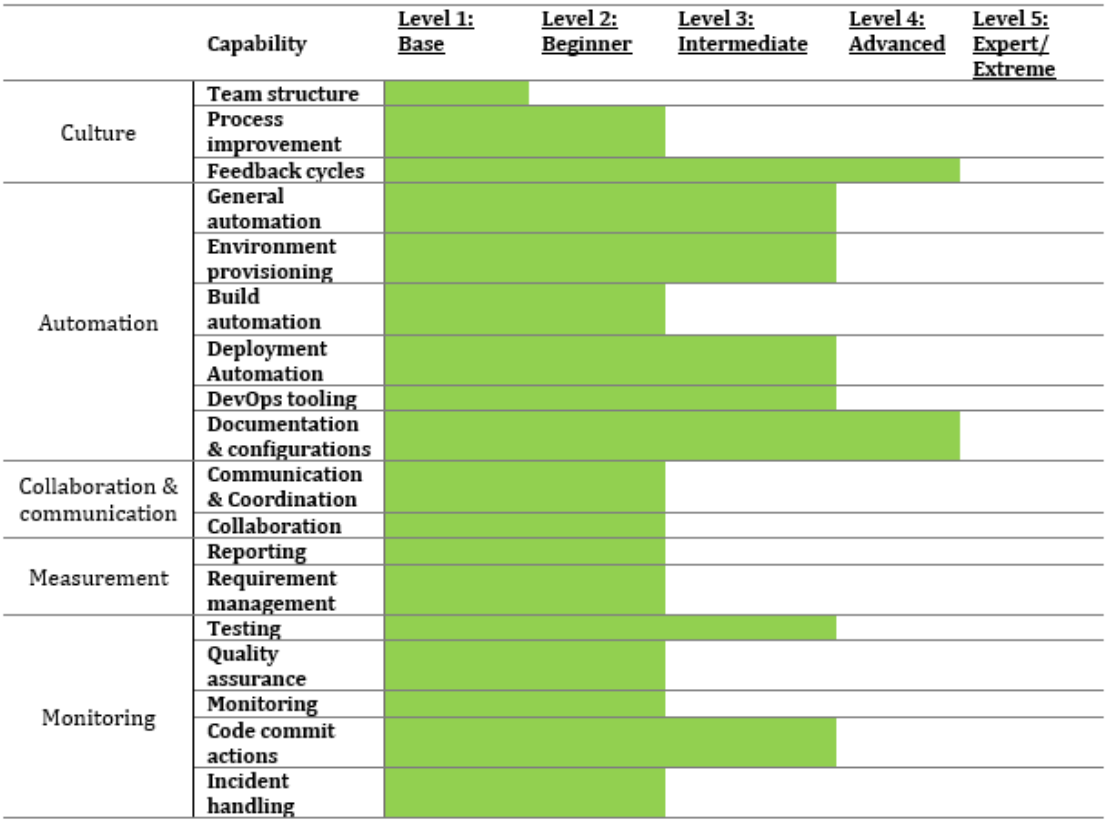

Figure 6.6: DevOps maturity assessment results case 6

By analyzing figure 6.6 most of the capabilities are at the beginner level. The only exception is the automation capabilities that mostly are level 3. The focus seems to be on the hard-skills of DevOps (automation) and not so much on the soft-skills. This could be by choice, but if not, then this leaves room for quite a bit of improvement that should lift the whole project into a better DevOps state.

6.1.7 Case 7

The project at organization F is a cloud based product that has been continuously improved for 4 years. About 120 people are involved spread across 6 teams. The biggest reason for adopting DevOps is to provide an environment in which rapid development can take place throughout the lifecycle. The results of the DevOps maturity assessment can be seen in figure 6.7

Figure 6.7: DevOps maturity assessment results case 7

Case 7 is the most mature case; this project has been focusing on DevOps since the start. This directly becomes clear from the maturity assessment, in which each capability has reached at least level 3 and the majority even level 4. This may be one of those projects that could be suitable for level 5, however, in that case, there are still some steps to be taken.

6.1.8 Case 8

This case is on a project that has only existed for about 9 months. This would indicate that the maturity is not expected to be very high. However, the project was started with the DevOps methodology in mind and by an organization with DevOps experience. The results of the assessment can be seen in figure 6.8

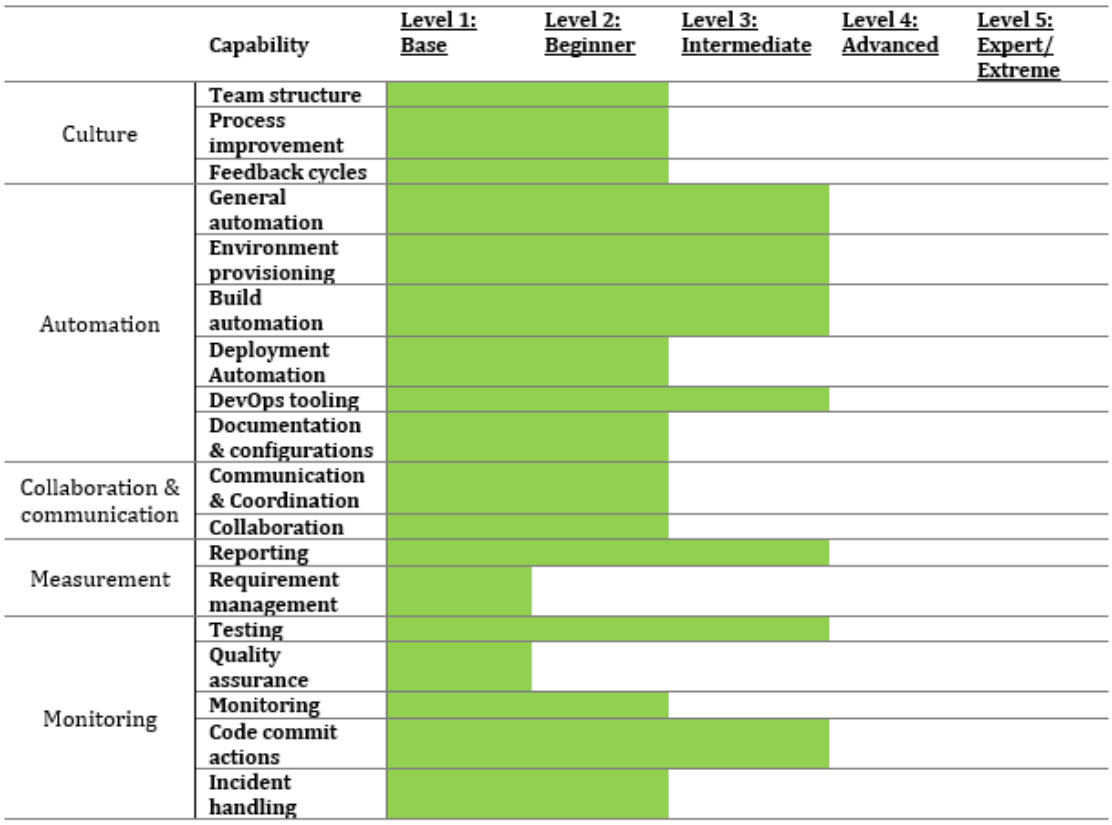

Figure 6.8: DevOps maturity assessment results case 8

The maturity of the project for case 8 differs from level 1 to level 3. The requirement management and quality assurance are at maturity level 1, which could be due to the age of the project. The level 3 capabilities are mainly focused around automation. Finally, the soft skills are overall level 2.

6.2 DISCUSSION OF CASE STUDY RESULTS: CROSS-CASE ANALYSIS

The case studies provided interesting results on their own basis. However, it is important to compare these results to find if any conclusions can be taken from them. Table 6.1 gives an overview of the average levels of each capability and the average deviation.

Table 6.3: Descriptive statistics of case studies

Capability	Minimum	Maximum	Average	St. Deviation
Team structure	1	3	1.63	0.92
Process improvement	2	3	2.75	0.46
Feedback cycles	2	4	2.88	0.83
Culture totals	1	4	2.42	0.93
General automation	2	4	3	0.76
Environment provisioning	1	4	3.13	1.13
Build automation	2	4	3.38	0.74
Deployment Automation	1	4	2.88	1.13
DevOps tooling	3	4	3.63	0.52
Documentation & configurations	2	4	3.25	0.71
Automation totals	1	4	3.21	0.85
Communication & Coordination	2	4	2.88	0.64
Collaboration	2	3	2.63	0.52
Collaboration & communication totals	2	4	2.75	0.58
Reporting	2	3	2.88	0.35
Requirement management	1	3	2.25	0.89
Measurement totals	1	3	2.56	0.76
Testing	2	4	2.88	0.64
Quality assurance	1	3	2	0.53
Monitoring	1	4	2.5	0.93
Code commit actions	3	4	3.38	0.52
Incident handling	2	4	3.13	0.99
Monitoring totals	1	4	2.78	0.85

The numbers can be biased because of the assessment taking place at a limited amount of organizations. Especially the capability 'team structure' as this was chosen by organization A to be kept at level 1 and this organization had four out of the eight assessments.

Another interesting observation is that the average on automation is higher than any of the other categories. This can probably be explained by the lack of investment into soft-skills of DevOps. This resulted mainly in investments into automation and thus possibly explaining the higher average of maturity level.

This assumption is reinforced by the high deviation in the culture category. As this category is only chosen to be implemented by a select number of case studies. Therefore, explaining the high deviation in this soft-skill category.

The highest standard deviation for capabilities is for environment provisioning and deployment automation. Both these could be more mature steps of automation. As first the build is automated, later followed by deployment and possibly by the environment. Therefore, it could be questioned if DevOps can only be at a specific maturity level if all capabilities are at that level. Or that these capabilities as automatic deployment should start at a higher DevOps maturity level to begin with.

The highest average capability is 'DevOps tooling'. This is a capability that was added during the DevOps expert interviews and is seeming to be considered very important and necessary to implement DevOps. To a certain extent it is true that without the tooling it is impossible to implement DevOps, however, it is the combination with the other capabilities that create the thriving environment.

Table 6.1 indicates that none of the capabilities were scored at a level 5 in any of the study cases. A possible explanation for this could be that more DevOps implementation will happen in the future, or that it does not fit the needs of the projects. This would confirm the results from the expert opinions and literature, where it was expected that the highest level would only fit very specific projects.

When comparing the maturity of the case studies in relation to the achieved benefits by the DevOps implementation it becomes clear that the higher the maturity, the more benefits were described in the assessment. This could mean that a higher maturity level results in achieving more DevOps benefits, which can motivate organizations to attempt to achieve a higher DevOps maturity level. However, it could be possible that some projects have more complex products and therefore, reach less benefits. Unfortunately, not enough information on the projects is known to make any definite conclusion. Nevertheless, it is interesting to consider the possibility of a higher maturity level contributing to more DevOps advantages. More details on this matter and other possible reasons for organizations to adapt the DevOps maturity model are described in section 7.3.

Finally, the overall deviations of the average levels appear to be quite small. This indicates that the implementation of DevOps in the various projects and across the organizations does not have a significant difference. This would confirm that a certain ideal environment consists. However, it could be possible that this dif-

fers per culture. For example, the team structure scored consistently low, as most projects used the traditional team structure organized around a skillset. Keeping Development and operations teams separate even though frequent communication exists between the teams.

6.3 SUGGESTIONS FOR IMPROVEMENT

In this section the possible improvement to the DevOps maturity model V3 will be discussed based on the results from the case study.

Firstly, it would be very helpful to have more guidance in the model. For example, by stating what is considered the best fit for most projects. This could be achieved by doing further research into existing DevOps projects around the world, which would also contribute to the existing research gap.

Secondly, the emphasis on the soft-skills of DevOps has become clear throughout the literature review. However, the expert interviews and case studies have not made the priority for these skills clear in a practical situation. It would, therefore, be recommended to revisit these categories in the DevOps maturity model and possibly adjust those to specific organizations and regions. However, before such a change can be made it is necessary to do more research.

Finally, level 5 did not occur in any of the case studies. Therefore, it would be interesting to research for which projects it would be sensible to try to achieve this level of maturity. If the subset of projects, for which this level is useful, is extremely small, it could be considered an unnecessary addition to the DevOps maturity model and possibly be removed in future iterations.

7

RECOMMENDATIONS FOR ORGANIZATIONS' USE OF THE DEVOPS MATURITY MODEL

The previous section showed that it is possible to apply the DevOps maturity model to a DevOps environment of an organization.However, the uncertainty on which steps to take to apply the maturity model can deter potential users. This chapter is focusing on helping organizations adopt to the DevOps maturity model. A general approach is described in section 7.1.

Research by Staples et al. shows that organizations can be reluctant to adopt CMMI. The reasons for not adopting CMMI should be addressed for the DevOps maturity model to prevent that the same reasons will impact the adoption of the DevOps maturity model. The first two reasons that companies do not apply maturity models in their organizations is that it is considered too costly and taking too much time. These concerns will be addressed in section 7.2. Finally, another reason was that organizations have already adopted other methods for process improvement, this will be addressed in section 7.3.

7.1 HOW TO APPLY THE DEVOPS MATURITY MODEL

The maturity model proposed in this thesis could be used in at least two ways: first to assess where a Devops organization stands in terms of DevOps implementation and secondly to improve upon those results. To be able to use the model, the organization needs to have a person knowledgeable of DevOps implementation to understand the necessary capabilities for each maturity level.

Secondly, the process of applying the model is iterative. The assessments should be done regularly, because DevOps implementation can change rapidly. However, because of the iterative nature it should be easy to apply the model. The DevOps maturity model has been given check-boxes for each item that can be useful to progress further. This can easily be maintained and provide a good overview for what is done and what is not, at any moment. It is also possible to transfer the model into a questionnaire, as we have done for the case studies in chapter 6. This provides the flexibility to the organization to change the assessment fitting to the requirements needed for their DevOps environment. Also the model is a continuous maturity model, as described before, which means that certain capabilities can be removed from the DevOps assessment if it not fitting for the project.

Filling in the DevOps maturity model will result in a completed model that provides insight in the current state of affairs. This is an important start for a company to decide which capability to invest in. For example, the results of case study 1 to 4 in this research shows a clear lacking in team organization. Knowing this the organization can start to restructure the teams, to be more agile and thus improving on the maturity level.

7.2 LIMITING COST AND TIME

The first two reasons that can prevent the adoption of the DevOps maturity model are the cost and time involved by applying the model. The time necessary for applying this model can be reduced by creating an re-usable assessment form for determining the maturity levels on each capability. This has been described in the previous section.

Another option for reducing the time needed for performing the DevOps maturity assessment is by selecting the capabilities that are most relevant for the organization. The maturity model is a continuous fixed level maturity model, which means that no set sequential path exists for reaching a certain overall maturity. Organizations can cherry-pick which capabilities they want to apply and remove others that are not relevant for them from the assessment form and thereby saving time.

The research of Staples et al. was based on the difficulties of a sales organization selling CMMI appraisal and improvement services. The organizations approached by sales were finding the costs too high. However, with the help of an assessment form it is possible to perform these maturity assessments in-house. This will be considerably cheaper than hiring another organization for doing the assessment.

7.3 BUSINESS CASE FOR ADOPTING TO THE DEVOPS MATURITY MODEL

The final reason given by organizations for not adopting to the CMMI model was that other models and process changes had already been adopted. Therefore, it is wise to look at why an organization should change to the DevOps maturity model.

It is difficult to suggest a standard level of maturity that provides the best balance in achieving DevOps benefits and implementation costs. This will also differ per organization and per project. The same is true for trying to quantify the cost of reaching the next maturity level for a specific capability and what benefits are expected.

However, it did become clear, from the case studies, that projects with a higher maturity level got more of the benefits that are attributed to DevOps in comparison to the lower maturity level projects. An assumption can be made that for achieving the DevOps benefits it is necessary to progress in the DevOps maturity model. If the organization wants to achieve those benefits, then they should adopt the DevOps maturity model to their project to guide them through achieving the maturity level that fits.

8 DISCUSSION

As it became clear from the literature review, hardly any processes and methods in adopting a DevOps maturity model to an organization were available. Therefore, it was aimed to create a comprehensive and validated DevOps maturity model in this research. The main contribution is to fill in the gap that was found in the literature, however it does not come without any limitations. These limitations are detailed first, followed by the contributions to research and practice.

8.1 DESIGN CHOICES

During the development of the DevOps maturity model, several design choices had to be made. Key choices are described in the following sections.

8.1.1 Selection of continuous fixed level maturity model

As described in the background, two types of maturity models exist. First, the fixed level maturity model, which can be split into staged maturity models and continuous maturity models. Second, the focus area maturity model. For the DevOps maturity model the fixed level continuous maturity model was chosen. The first reason for choosing the fixed level maturity model, is the widely known application of this model throughout organizations' hierarchical levels. This popularity stems from the fact that the predecessor 'CMM' (Capability Maturity Model) has been applied in many information system areas since 1993 [23]. The focus area maturity model was only introduced in 2010 [25]. The amount of supporting literature for CMMI is high, and one of the common issues with maturity models discovered in a literature review by Poeppelbuss et al. is that the foundation of research from existing literature was missing [58]. We wanted to prevent that by choosing the fixed level maturity model. Also, we though that choosing a lesser known model could give issues in the research and in practical situations. This was confirmed during the first interview with the experts, as the CMMI based (fixed level) maturity model was the only model that they could recall.

The option to choose specifically the continuous version of the maturity model was that DevOps implementation can differ greatly per organization and that if they are lacking in one area then it does not mean that the whole DevOps implementation is immature. Also, the staged maturity model assumes that a standard sequence of improvements exists, which is not the case for DevOps. This was confirmed during the case studies in which it became clear that many differences exist between the various capabilities' levels, but most of the DevOps implementations were considered successful by the organizations.

8.1.2 Selection of levels for maturity model

The levels chosen for the DevOps maturity model started of as the exact same as used in the CMMI. These were eventually changed to different terms, based on the responses given in the expert interviews. This increased the understandability and described the levels better in the context of DevOps. However, no comments were made regarding the division of DevOps maturity into five levels. Would a different scale be better suitable for DevOps?

Not much evidence currently exists on the proper leveling of the gradual growth in implementing DevOps. In the literature review it became clear that other researchers mostly chose for 5 levels [43], [44], [46]–[48], but also for 4 [45]and 10 level [2] models. However, it is unclear which scale would be best suitable. Researching the possibility of different levels for the growth was outside of the scope of this research. Therefore, it was chosen to keep the same levels as the CMMI model, as well as most other researchers have done. The most important reason for maintaining the levels is the same as described in the previous section. The CMMI model is popular and widely known.

8.2 LIMITATIONS

This section describes the limitations of this research. First the limitations regarding the maturity model are outlined, followed by the limitations regarding the case study.

8.2.1 DevOps maturity model

The literature research that resulted in a DevOps maturity model had a scope specific to existing maturity models. It is however possible that other capabilities could have been found in literature that are now missing in the maturity model. It was also chosen to only select the capabilities for the maturity model that were mentioned in at least two pieces of literature. This was done to prevent using capabilities that were not fully backed up by the existing literature, however this does not mean that no useful capabilities exist in between those that were excluded. As a result, only a part of all capabilities from literature have been captured. The chosen literature only came from a limited number of databases, it is possible that other databases could have provided more literature. We did our best to consult all results from the used queries, however some were inaccessible due to access restrictions. Finally, it is not guaranteed that the queries chosen, sufficiently represented the goal to retrieve all literature available on DevOps maturity models.

The second source for creating the DevOps maturity model was expert interviews. The first limitation is the number of interviews, only four experts from three different organizations were found with enough experience and knowledge on DevOps to contribute meaningfully to the DevOps maturity model. Originally five expert interviews were arranged, however during one of the interviews it became clear that the participants did not have any knowledge on DevOps, even

though this was communicated differently beforehand. Therefore, this interview had to be discarded.

The experts all originated from Ho Chi Minh City, which places a large limitation on the results as it could be too restrictive because of similar backgrounds. It was tried to avoid this by selecting the participants from both a research and practical background, however it cannot be fully avoided. This is due to culture being one of the cornerstones of DevOps and the organization culture can differ greatly between geographical regions. For example the cultures in organizations inside (South)East-Asia is more hierarchical in comparison to American/European organization cultures [38]. This could influence the close collaboration needed for DevOps, which is considered the core category for DevOps adoption by Luz et al [59]. Another example is in the fear of failure or 'losing face' in (South)East-Asian companies [60] which could be an obstacle in the teamwork and evaluation principles of DevOps. Therefore, this could limit the generalizability of what was found during the interviews to only (South)East-Asian organizations.

The difficulty of finding suitable experts arose as well from the infancy of DevOps in Vietnam. which places a large limitation on the results as it could be too restrictive because of similar backgrounds. It was tried to avoid this by selecting the participants from both a research and practical background, however it cannot be fully avoided. The difficulty of finding suitable experts arose as well from the infancy of DevOps in Vietnam.

After combining both sources into one DevOps maturity model, it was chosen to do a group validation. The reason for this is that the consensus among the participants could result in a better validation of the model. However, due to time constraints and scheduling conflicts we were unsuccessful to get the experts together. Therefore, it was chosen to interview the experts separately. Doing this separately can be considered a limitation of the model validation.

The resulting DevOps maturity model provides several levels of DevOps implementation. However, something that could be considered is that from a first look at the model it is easy to assume that the highest level is the best. However, this is very dependent on the organization that is using DevOps. It could therefore be made clearer that the suitable level can differ per organization, as this does not come clear from the model itself. However, this would be something that can be addressed in future research.

8.2.2 Case study

Case studies were performed to test the practical usage of the model. Semi-structured Interviews would have been preferred to questionnaires, because of the possibility to get more in-depth in the questioning and being able to respond to the interviewee's responses. Unfortunately, most of the participants requested a digital questionnaire, because of limitations in their spoken English. This presented difficulty in assessing the companies DevOps maturity, however the combination of the data from questionnaires, asking sub-sequent questions if uncertainties arose and observing the projects made it possible to create a more

complete result in at least four of the cases at organization A. The other case studies were still very useful even without observations.

An additional limitation is the amount of information that was shared regarding the case study projects. The participants were conscious about sharing as much information as their company policy allowed. However, this was mostly limited to the number of employees, duration of the project, number of teams and, mostly, a vague description on what was being developed. All other information requested was considered proprietary. This, in turn, limited the usability of the case studies to extract more meaningful conclusions.

Another limitation that arose in the application of the model was that the different capabilities can vary significantly. For example, one of the companies had a high maturity in most of the automating capabilities but were lacking in the culture aspect. This corresponded with observing the difficulty in acquiring funding and project time for improving the soft aspects of DevOps.

Other limitations are regarding the number of participants, as more participants could have provided more input for evaluation of the model. Another limitation is the geographic location, because this research was solely performed Ho Chi Minh City, Vietnam. This poses a limitation to the generalizability of the case study results, which will be discussed in Appendix E.

Furthermore, as Yin [57] suggests, a central question in case study research is related to the extent to which our results could be observable in other similar but different settings. For example, similar organizations in other regions of Vietnam, or in other countries in the same geographic region. Of course, more research is needed to confirm if our findings would be similar to observations in other organizations and project contexts. However, following Scheepers and Seddon [61] and Wieringa and Daneva [62], we could possibly think that there would exist similarities between our findings and observations in other organizations if these other organizations are similar to those in which we did the case studies. We think this is possible because similar organizations might have similar mechanisms in place that lead to the use of similar practices. As described before, these mechanisms can be different across cultures. However, the cultures are not expected to differ much inside the geographical area, and this could mean that the interviews performed with in similar jobs throughout (South)East-Asia, for example Cambodia, China etc., would have provided similar results.

8.3 CONTRIBUTIONS TO RESEARCH AND PRACTICE

8.3.1 Contributions to research

As became clear from the literature used in this research there is still a gap regarding DevOps literature. More specifically, most DevOps models exists in grey literature and are used in a specific company. This scientific research contributes in three ways to existing research. First, a comprehensive overview of existing DevOps maturity models has been presented. Second, the existing literature has been combined with empirical research to create a comprehensive

DevOps maturity model which can be used for future research. Third, research in (South)East-Asia on DevOps is even more scarce and this research adds to the body of knowledge on DevOps in a particular region. It provides empirical findings from research on companies located in Vietnam.

8.3.2 Contributions to practice

This research provides a practical DevOps framework that can be applied by DevOps practitioners to establish the maturity of DevOps inside their organizations. The participants in this research have taken a positive stance regarding such a framework, which could mean that this is something that can be used in practice. However, the need to grow in maturity has been questioned by participants as some measures that have to be taken can be quite expensive for minimal return. Therefore, it remains important to try to fit this (theoretical) framework to the practice of the specific organization.

Another usage of the model that was voiced by multiple participants, is that this model can be useful in explaining to management where further investment is necessary. Frequently management does not know the need to improve the soft aspects (culture, communication, collaboration) of DevOps and therefore limits funding for this. The participants were hopeful that this model can help change the mind and secure extra funding from management.

9 CONCLUSION AND FUTURE WORK

This section will provide answers to the research questions, by recapitulating the chapters in this report. The answering of the research questions is followed by an overall conclusion. This chapter is concluded by suggesting future work that can build on this research and other gaps to fill in the DevOps area.

9.1 RESEARCH QUESTIONS

The study aimed to create a suitable maturity framework to assess and improve DevOps environments in various organizations. To achieve this, four sub-questions were posed. Each of these sub-questions will be answered before discussing how we reached the main research objective.

SQ1. What models on maturity in DevOps exist and which are suitable to use for creating a comprehensive model applicable to all businesses working with DevOps?

In order to answer this question a thorough literature review was performed. This review resulted in an overview of existing maturity models. Most of these models were based on the CMMI model and were limited in their granularity. One other model existed which had a fine granularity, however this was not based on the CMMI model and could therefore have limited success in organizations due to being unfamiliarity. As no all-around DevOps maturity model seemed to exist it was decided to compare all capabilities mentioned in the literature on DevOps maturity models. This comparison resulted in a list of capabilities and with these capabilities a new CMMI based model was created with a high granularity to better assist companies in adapting to a DevOps environment. A full overview of this process can be viewed in chapter 4.

SQ2. What do experts in the DevOps field see as the capabilities necessary for successful DevOps implementation?

The second sub-question was answered by interviewing experts in the DevOps field. These interviews resulted in changing and adding capabilities to the DevOps maturity model. For example, DevOps tooling was mentioned to be very important as well as continuous monitoring and changing monolith structure of software applications to microservices. Also, many of the capabilities that already existed in the model were confirmed during the interviews. The full process of expert interviews can be found in chapter 5.2

SQ3. How do experts in the DevOps field evaluate the maturity model and what improvements should be done based on their responses?

Answering the third sub-question was done by means of expert opinion. The experts were inquired about how they would think that the DevOps maturity model would interact with real-life scenarios. One of the biggest changes that resulted from this is the change to the naming of the maturity levels. As well as adding the capability for incident handling. The complete overview of the

validation of the model can be seen in chapter 5.3 Based on the feedback it was chosen to adopt the model to its final version that can be viewed in Appendix C. The

SQ4. What is the DevOps maturity level of the organizations used for the case studies and how can this be explained?

The case studies that were assessed with the DevOps maturity model resulted in various maturity levels. Not only in-between the organizations but also in-between the capabilities. For example, in most case studies it became clear that automation has more priority than the soft-skills that fits a DevOps environment. It became clear that management does not always see the necessity of improving communication or culture by taking away hierarchy. A full description of the case studies can be found in chapter 6

SQ5. How did the DevOps maturity model perform in this case study and what improvements should be done based on these findings?

The DevOps maturity model proved to be very useful in assessing the level of DevOps implementation in an organization. As well as in helping management layers understand what is necessary to have a better DevOps environment. This already helped one company to secure more investment into the culture and to not only focus on the automation part. This could help improve the organization in its DevOps capabilities in the future and thus exploiting the benefits of DevOps more. The model could be improved by adding more information about which level is suitable for most projects and if it is necessary to keep level 5 as a part of the model if this level does not seem suitable for many projects.

RO. What is a suitable maturity framework that allows organizations to assess and improve their DevOps environment?

The sub-research questions (SQ1 to SQ5) resulted in a suitable maturity framework that allows organizations to assess and improve their DevOps environment, thereby satisfying the research objective that was set out at the beginning of this research. The proposed maturity framework was evaluated by means of expert interviews and by eight case studies in diverse organizations.

9.2 KEY LESSONS LEARNED AND FINDINGS

This research was meant to fill a gap that was found during the literature review (chapter 4). However, the resulting model seems to also fit practical situations. Therefore, both practitioners and researchers are helped with this report. Besides the DevOps maturity model, other findings have been found throughout this research. The most important findings of this research are described in the following section.

1) The focus is mostly on the automation part of DevOps

In the case studies, it became clear that the automation maturity was always ahead of the soft-skills of DevOps. This could be due to many reasons, as the number of projects considered is an extremely small subset of all projects in the

world, and all projects were based in the same city. However, it is interesting nonetheless and could indicate the need for adjustments in DevOps to different cultural regions.

2) There is no fitting maturity level that fits all projects and organizations

The maturity levels in the model can be quite specific to certain projects. For example, in the case studies, it became clear that none of the projects had reached maturity level 5. This indicates that there may be a general recommendation on a maturity level that fits most projects, however, there is no maturity level that would fit all projects.

3) DevOps has a different meaning to different people

This is more a generalization of the second key finding. As has become clear by the interviews, DevOps has a different meaning to different people. Also, the literature was quite dispersed on what entails DevOps. For example, some think it should only be in relation to Development and Operations, whereas others think it should impact the whole organization.

9.3 FUTURE WORK

This study has provided an overview of existing DevOps models and build upon this to create a comprehensive model based on CMMI with a fine granularity. However, this research is not complete and can be taken a further look at. These opportunities will be described in this section

The list of DevOps maturity models that was compared in the literature review can be used by other researchers as a start for developing an understanding of DevOps maturity models. In this case the models have been compared and if a capability existed in more than one article, then this would be used in the DevOps maturity model. Follow-up research could take a more in-depth analysis and establish if any important capabilities have been missed by taking this approach.

The DevOps model created in this research can be used by other researchers to further build upon. This can be achieved by doing more empirical research, which will not only contribute to DevOps maturity models but to DevOps research in general, which is still a lacking area in scientific research.

Finally, through the case studies, it has become clear that there is no fitting maturity level for all organizations. This can be influences by many matters, the organization, product complexity, role of the teams and team member capabilities, just to name a few; however, it may be possible to establish a general 'rule of thumb' that will fit most companies. This level can differ per capability, so an in-depth look would have to be taken into this.

10

BIBLIOGRAPHY

[1] C. Development, “CMMI ® for Development, Version 1.3," no. November, 2010.

[2] R. de Feijter, S. Overbeek, R. van Vliet, E. Jagroep, and S. Brinkkemper, “DevOps competences and maturity for software producing organizations," *Lect. Notes Bus. Inf. Process.*, vol. 318, pp. 244–259, 2018.

[3] I. Kornilova and Medium, “DevOps is a culture, not a role!," 2017. [Online]. Available: https://medium.com/@neonrocket/devops-is-a-culture-not-a-role-be1bed149b0.

[4] D. Stahl, T. Martensson, and J. 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)*, 2017, pp. 440–448.

[5] L. E. Lwakatare, P. Kuvaja, and M. Oivo, “Relationship of DevOps to Agile, Lean and Continuous Deployment," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10027 LNCS, 2016, pp. 399–415.

[6] P. Rodríguez *et al.*, “Continuous deployment of software intensive products and services: A systematic mapping study," *J. Syst. Softw.*, vol. 123, no. 2017, pp. 263–291, Jan. 2017.

[7] F. M. A. Erich, C. Amrit, and M. Daneva, “A qualitative study of DevOps usage in practice," *J. Softw. Evol. Process*, vol. 29, no. 6, p. e1885, Jun. 2017.

[8] J. Sharp and J. Babb, “Is Information Systems Late to the Party? The Current State of DevOps Research in the Association for Information Systems eLibrary," in *Twenty-fourth Americas Conference on Information Systems*, 2018.

[9] G. Ghantous and A. Gill, “DevOps: Concepts, Practices, Tools, Benefits and Challenges," *PACIS 2017 Proc.*, vol. 9132, p. 96, 2015.

[10] A. Balalaie, A. Heydarnoori, and P. Jamshidi, “Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture," *IEEE Softw.*, vol. 33, no. 3, pp. 42–52, May 2016.

[11] C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano, “DevOps," *IEEE Softw.*, vol. 33, no. 3, pp. 94–100, May 2016.

[12] J. Humble and J. Molesky, “Why Enterprises Must Adopt DevOps To Enable Continuous Delivery," *Cut. IT J.*, vol. 24, no. 8, pp. 6–12, 2011.

[13] L. Lwakatare, M. Oivo, and P. Kuvaja, “An exploratory Study of DevOps: Extending the Dimensions of DevOps with Practices," in *ICSEA 2016*, 2016.

[14] L. E. Lwakatare, P. Kuvaja, and M. Oivo, “Dimensions of DevOps," in *Dimensions of DevOps*, 2015, pp. 212–217.

[15] M. Virmani, “Understanding DevOps & amp; bridging the gap from continuous integration to continuous delivery," in *Fifth International Conference*

on the Innovative Computing Technology (INTECH 2015), 2015, no. Intech, pp. 78–82.

[16] M. Shahin, "Architecting for DevOps and Continuous Deployment," in *Proceedings of the ASWEC 2015 24th Australasian Software Engineering Conference on - ASWEC ' 15 Vol. II*, 2015, pp. 147–148.

[17] M. Hüttermann, "Infrastructure as Code," in *DevOps for Developers*, Berkeley, CA: Apress, 2012, pp. 135–156.

[18] E. Shamow, "Devops at Advance Internet: How We Got in the Door," *Cut. IT J.*, vol. 24, no. 8, pp. 13–18, 2011.

[19] S. K. Bang, S. Chung, Y. Choh, and M. Dupuis, "A grounded theory analysis of modern web applications," in *Proceedings of the 2nd annual conference on Research in information technology - RIIT '13*, 2013, p. 61.

[20] D. G. Feitelson, E. Frachtenberg, and K. L. Beck, "Development and Deployment at Facebook," *IEEE Internet Comput.*, vol. 17, no. 4, pp. 8–17, Jul. 2013.

[21] C. G. Von Wangenheim *et al.*, "Systematic Literature Review of Software Process Capability / Maturity Models," no. May, 2010.

[22] L. A. Lasrado, R. Vatrapu, and K. N. Andersen, "Maturity Models Development in IS Research: A Literature Review," *Proc. 38th Inf. Syst. Res. Semin. Scand. (IRIS 38)*, vol. 6, no. August 9-12, pp. 1–12, 2015.

[23] M. C. Paulk, C. V Weber, and M. B. Chrissis, "The Capability Maturity Model" for Software'," *IEEE*, pp. 1–26, 1993.

[24] J. van Hillegersberg, "The Need for a Maturity Model for Maturity Modeling," *Art Struct.*, pp. 145–151, 2019.

[25] M. van Steenbergen, R. Bos, S. Brinkkemper, I. van de Weerd, and W. Bekkers, "The Design of Focus Area Maturity Models," 2010, pp. 317–332.

[26] J. Becker, R. Knackstedt, and J. Pöppelbuß, "Developing Maturity Models for IT Management," *Bus. Inf. Syst. Eng.*, vol. 1, no. 3, pp. 213–222, 2009.

[27] A. Tarhan and O. Turetken, "Business Process Maturity Models: A Systematic Literature Review Business Process Maturity Models: A Systematic Literature Review," no. November 2017, 2016.

[28] F. McGarry, S. Burke, and B. Decker, "Measuring the impacts individual process maturity attributes have on software products," in *Proceedings Fifth International Software Metrics Symposium. Metrics (Cat. No.98TB100262)*, pp. 52–60.

[29] R. Dion, "Elements of a process-improvement program (software quality)," *IEEE Softw.*, vol. 9, no. 4, pp. 83–85, Jul. 1992.

[30] J. D. Herbsleb and D. R. Goldenson, "A systematic survey of CMM experience and results," in *Proceedings of IEEE 18th International Conference on Software Engineering*, 1996, pp. 323–330.

[31] J. Herbsleb, A. Carleton, J. Rozum, J. Siegel, and D. Zubrow, “Benefits of CMM-Based Software Process Improvement: Initial Results," 1994.

[32] D. R. Goldenson and D. L. Gibson, “Demonstrating the Impact and Benefits of CMMI ®: An Update and Preliminary Results," no. October, pp. 7–10, 2003.

[33] D. Gibson, D. Goldenson, and K. Kost, “Performance Results of CMMI-Based Process Improvement," 2006.

[34] H. Glazer, J. Dalton, D. Anderson, M. Konrad, and S. Shrum, “CMMI or Agile: Why Not Embrace Both!," no. November, 2008.

[35] L. F. Chagas, D. D. de Carvalho, A. M. Lima, and C. A. L. Reis, “Systematic Literature Review on the Characteristics of Agile Project Management in the Context of Maturity Models," 2014, pp. 177–189.

[36] F. Selleri *et al.*, “Using CMMI together with agile software development: A systematic review," vol. 58, pp. 20–43, 2015.

[37] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, “A Design Science Research Methodology for Information Systems Research," *J. Manag. Inf. Syst.*, vol. 24, no. 3, pp. 45–77, Dec. 2007.

[38] C. Flora Hung, “Cultural influence on relationship cultivation strategies: Multinational companies in China," *J. Commun. Manag.*, vol. 8, no. 3, pp. 264–281, Jul. 2004.

[39] P. Alpar, “Turnover intentions of employees of information technology outsourcing suppliers in Vietnam."

[40] D. Tranfield, D. Denyer, and P. Smart, “Towards a Methodology for Developing Evidence-Informed Management Knowledge by Means of Systematic Review," *Br. J. Manag.*, vol. 14, no. 3, pp. 207–222, Sep. 2003.

[41] V. Garousi, M. Felderer, and M. V. Mäntylä, “The need for multivocal literature reviews in software engineering," *Proc. 20th Int. Conf. Eval. Assess. Softw. Eng. - EASE ’16*, pp. 1–6, 2016.

[42] V. Garousi, M. Felderer, and M. V. Mäntylä, “Guidelines for conducting multivocal literature reviews in software engineering," *Inf. Softw. Technol.*, vol. 000, no. May, 2017.

[43] S. I. Mohamed, “DevOps Shifting Software Engineering Strategy Value Based Perspective," *IOSR J. Comput. Eng. Ver. IV*, vol. 17, no. 2, pp. 2278–661, 2015.

[44] I. Bucena and M. Kirikova, “Simplifying the DevOps Adoption Process Challenges of DevOps Adoption."

[45] P. Bahrs, “Adopting the IBM DevOps approach for continuous software delivery Adoption paths and the DevOps maturity model," 2013.

[46] P. Bostrom, A. Rehn, and T. Palmborg, “Continuous Delivery Maturity Model InfoQ," 2014.

[47] IBM Corporation and E. Minick, “Continuous Delivery Maturity Model," 2014.

[48] Forrester Consulting, “Continuous Delivery: A Maturity Assessment Model," 2013.

[49] A. D. Nagarajan and S. J. Overbeek, “A DevOps Implementation Framework for Large Agile-Based Financial Organizations," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 11229 LNCS, pp. 172–188, 2018.

[50] G. Rong, H. Zhang, and D. Shao, “CMMI guided process improvement for DevOps projects," in *Proceedings of the International Workshop on Software and Systems Process - ICSSP '16*, 2016, pp. 76–85.

[51] R. J. Wieringa, *Design science methodology: For information systems and software engineering*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014.

[52] T. Wengraf, *Qualitative Research Interviewing*. 1 Oliver's Yard, 55 City Road, London England EC1Y 1SP United Kingdom: SAGE Publications, Ltd, 2001.

[53] J. Saldana., *Coding manual*. 2013.

[54] M. B. Miles and M. A. Huberman, “Matthew B. Miles, Michael Huberman - Qualitative Data Analysis_ An expanded Sourcebook 2nd Edition (1994).pdf." p. 338, 1994.

[55] W. H. D. Lone and E. R. M. Lean, “The DeLone and McLean Model of Information Systems Success: A Ten-Year Update," *J. Manag. Inf. Syst.*, vol. 19, no. 4, pp. 9–30, 2003.

[56] S. T. March and G. F. Smith, “Design and natural science research on information technology," *Decis. Support Syst.*, vol. 15, no. 4, pp. 251–266, Dec. 1995.

[57] R. Yin, *Case Study Research and Applications*. 2017.

[58] J. Poeppelbuss, B. Niehaves, A. Simons, and J. Becker, “Maturity Models in Information Systems Research: Literature Search and Analysis," *Commun. Assoc. Inf. Syst.*, vol. 29, 2011.

[59] W. P. Luz, G. Pinto, and R. Bonifácio, “Building a collaborative culture," pp. 1–10, 2018.

[60] H. von Weltzien Hoivik, “East Meets West: Tacit Messages about Business Ethics in Stories Told by Chinese Managers," *J. Bus. Ethics*, vol. 74, no. 4, pp. 457–469, Sep. 2007.

[61] P. B. Seddon and R. Scheepers, “Towards the improved treatment of generalization of knowledge claims in IS research: Drawing general conclusions from samples," *Eur. J. Inf. Syst.*, vol. 21, no. 1, pp. 6–21, 2012.

[62] R. Wieringa and M. Daneva, “Six strategies for generalizing software engineering theories," *Sci. Comput. Program.*, vol. 101, pp. 136–152, 2015.

A

STRUCTURED LITERATURE REVIEW

A.1 SEARCH PROTOCOL

Table A.1: Summary of the Search strategy

Academic databases search	Scopus
	Web of Science
	IEEE Xplore
	ACM Digital Library
	AIS Electronic library
	Springer Database
Other data sources	Sans Database search via Google (nonacademic resources)
Target items	Journal papers
	Workshop papers
	Conference papers
	Industry/professional contributions
	Grey literature (white papers)
Search applied to	Title
	Abstract
	Keywords
	Full-text
Language	Papers written in English
Publication period	Until January 2019

Table A.2: Inclusion and exclusion criteria

Inclusion	**Exclusion**
Papers with DevOps maturity models	Papers about DevOps tools
Papers linking DevOps to CMMI	Papers with only a state of literature
	Paper must be in English

A.2 SEARCH RESULTS

Search in academic databases

Search key: “devops" AND (CMMI OR maturity)
Search conducted 05.02.2019

Table A.3: Search results in academic databases

Unique results	62
After filtering on title and English language	25
After filtering on abstract	12
After filtering on full-text	4
Adding forward & backward references	8

Search in grey literature

Search key: site:https://www.sans.org/reading-room filetype:pdf “devops" AND (CMMI or maturity)
Search conducted on 10.02.2019

Table A.4: Search results search in grey literature

Unique results	32
After filtering on title and English language	4
After filtering on abstract	2
After filtering on full-text	0
Adding forward & backward references	0

A.3 SUMMARIES OF THE SELECTED PAPERS

The papers that were selected are summarized in table A.5. First, the type of the paper is indicated, five papers were conference papers (C) and one white paper (W). As well as a short summary of the paper. Unfortunately, no viable results were found from the search in grey literature, however, four white paper was added as a backward reference from the search in academic databases.

Table A.5: Summary of selected papers for literature review

Paper	Type	Description
[50]	C	The paper reports a case study that evaluated the feasibility of CMMI models for guidance in process improvement for DevOps projects. Concluding that CMMI could be taken as a design foundation for maturity models in DevOps context.
[43]	C	Introduces a new DevOps maturity model based on CMMI maturity model as well as a model that helps in transforming the software development life cycle to adapt it with the DevOps strategy. Which leads to the acceleration of the delivery of application changes.
[44]	C	The paper reports on research results in facilitating DevOps in small enterprises. A DevOps maturity model is proposed based on five levels of maturity in four enterprise areas: technology, process, people and culture.
[45]	W	Outlines four paths for adopting/improving continuous software delivery in an organization: planning and measuring; developing and testing; releasing and deploying; monitoring and optimizing. Also, a practice-based DevOps maturity model is presented to help assess current practices, measure improvement and define a roadmap.
[46]	W	Aims to give structure and understanding of the key aspects that need to be considered in adopting continuous delivery by providing a continuous delivery maturity model
[47]	W	Presents a five-level maturity model, comparable to the CMMI model (albeit with different names for the levels). Followed by a more detailed description of the building, deploying, testing and reporting aspect of continuous delivery in an enterprise.
[48]	W	Companies are looking to prioritize innovation through developing software service, however the rate of software releases is not high enough to satisfy the business leaders. Only a few IT organizations can perform the needed regular continuous delivery practices. A cause that prohibits other organizations is the corporate culture and development process immaturity.
[2]	C	Proposes a DevOps maturity model, which can also be used for adopting DevOps. This is based on researched capabilities needed for successful DevOps implementation
[49]	C	Presents a conceptual framework on the implementation of DevOps in large-scale financial organizations. Practitioners have validated the framework, mainly to educate people in their organizations, and it has been successfully applied in real organizations as a case study.

B

CAPABILITIES OVERVIEW LITERATURE AND INTERVIEWS

This appendix shows the unfiltered list of capabilities from literature, one column is added for synthesizing it with results from the interviews (IV). The table is ordered to start with the capabilities that were mentioned in both the literature and interviews, followed by the capabilities found in multiple literatures and finally the capabilities only found in one piece of literature.

Table B.1: Comparison of literature on DevOps capabilities

Capabilities	[43]	[44]	[45]	[46]	[47]	[2]	[49]	IV
Manage environments through automation		X	X					X
Cross-functional teams with knowledge overlap		X		X		X	X	X
Shared KPIs between teams		X		X		X	X	X
Self-service infrastructure provision and deployment (IAC)			X	X	X	X		X
Quality gated commits					X	X		X
Continuous deployment	X	X		X	X	X	X	X
Automated (continuous) build creation		X				X		X
Continuous testing (automated recoverability)	X	X	X			X		X
Training and guidance (facilitators)							X	X
Leadership commitment							X	X
DevOps tooling		X						X
Change from monolith to microservices		X						X
Continuous monitoring		X						X
Processes focus on collaboration	X	X					X	
Constructive communication environment	X					X	X	
Continuous process improvement		X					X	
Rapid personnel feedback cycles		X					X	
Define release with business objectives			X			X		
Optimize to customer KPIs continuously			X			X		
Cross silo analysis				X	X			
Automated documentation generation		X				X		
Feature toggles				X		X		
100% coverage testing					X			
Knowledge sharing						X		
Trust and respect						X		
Branch & merge (feature toggles) (minimal branching)				X		X		

continued on next page

Configuration management					X	
Continuous architecture evolvement					X	
Non- hierarchical organizational structure						X
Enterprise wide agile practices						X
Open and trusted environment						X
Software configuration management		X				
External learning		X				
Continuous capability improvement		X				
Clear organization requirements		X				
Desired culture identified and ingrained		X				
Strategic innovation		X				
Performance management		X				
Quantitative project management		X				
Smart automatization to maximize throughput	X					
Optimized governance & self-adaptation	X					
Database management		X				
No rollbacks			X			
Verify expected business value				X		
Dynamic graphing and dashboards				X		
Measure to customer value			X			
Smart automatization to maximize throughput	X					
Improve continuously with development intelligence			X			
Automate problem isolation issue resolution			X			

		Maturity model V1				
	Capability	Level 1: Initial	Level 2: Managed	Level 3: Defined	Level 4: Measured	Level 5: Optimized
Culture	**Team Structure**	☐Isolated teams ☐Organized around skillset (dev/ops)	☐Teams structured around deliveries ☐One backlog per team	☐Teams structured around projects	☐structured around products ☐Cross-functional teams with knowledge overlap	☐Interdisciplinary teams and structured around KPIs ☐multi-skilled and flexible members
	Process improvement	☐All processes are manual or undefined	☐Processes and methods are defined intrateam	☐Adapting processes inter-team to make them more seamless	☐Adaption of methods and processes to all aspects of the organization	☐Continuous process improvement
	Feedback Cycle	☐Restricted feedback opportunities	☐Rapid intrateam feedback ☐Safe feedback environment	☐Rapid inter-team feedback	☐Frequent feedback on all processes	☐Continuous feedback process improvement
Automation	**General**	☐Ad-hoc delivery processes ☐Ad-hoc automation	☐Scheduled (delivery) processes	☐Automated (delivery) process ☐Standardized automation	☐Frequent delivery process ☐Continuous delivery	☐Smart automation to maximize throughput
	Environments (test, staging, production etc.)	☐Consolidated platform and technology ☐Manually provisioned infrastructure	☐Partly automatically provisioned infrastructure ☐Organize system into modules	☐Automatically provisioned infrastructure ☐Virtualization when applicable ☐standard process across all environments	☐Manage environments through automation ☐Full component-based architecture	☐Infrastructure As Code ☐All environment configuration is externalized and versioned
	Build automation	☐Versioned code base ☐Scripted builds ☐Scheduled builds ☐Dedicated build server	☐Build automation ☐Builds are stored ☐manual tag & versioning	☐Auto triggered build ☐automated tag & versioning ☐build once and deploy anywhere	☐Build metrics gathered, made visible and considered	☐Continuous build process improvement, e.g. faster feedback
	Deployment automation	☐Manual deployment ☐some deployment scripts exist	☐Mostly standardized deploys ☐Partly automated deployment	☐Non-production deployment automation ☐basic deployment pipeline for all environments ☐automated most of the database changes	☐Production deployment automation ☐Automated database changes ☐Zero downtime deploys ☐Parallel processing with multiple build machines ☐Continuous deployment to test	☐Collaboration between teams to manage risks and reduce cycle time ☐Zero touch continuous deployments ☐Continuous deployment to production ☐Feature toggles
	Documentation & configurations	☐Not available or is outdated	☐Up-to-date documentation and configurations	☐Regular validation of the documentation and configurations is performed	☐Documents are updated based on gathered experience and quality requirements	☐Automated generation of documentation and configurations

Maturity model V1 - continued						
	Capability	Level 1: Initial	Level 2: Managed	Level 3: Defined	Level 4: Measured	Level 5: Optimized
Collaboration & communication	**Communication & Coordination**	☐Indirect ☐Poor ☐Ad-hoc	☐Controlled ☐Facilitated ☐Actively managed	☐Standardized ☐Direct	☐Metrics for improving communication ☐Structured ☐Peer-to-peer ☐Constructive	☐Communication and coordination optimized ☐Community building Optimize communication
	Collaboration	☐ Ad-hoc collaboration	☐Active collaboration	☐Sharing pain between teams	☐Extended collaboration	☐Optimize collaboration
Measurement	**Reporting**	☐Manual reporting ☐Ad-hoc reporting ☐Only visible for the report runner	☐Scheduled quality reports ☐Tool generated reports ☐Latest reports always accessible ☐Report is visible for the whole team	☐Report history is available ☐Report is visible cross-silo	☐Report trend analysis ☐Collaboration based process measurement, which allows to identify bottlenecks and inefficacies	☐Cross silo analysis report
	Business objective focus releases	☐Manage department resources ☐Document objectives locally	☐Link objectives to release ☐Centralize requirement management	☐Plan and source strategically ☐Dashboard portfolio measures	☐Automation metrics exist to assess progress against business goals	☐Define release with business objectives ☐Measure and optimize to customer value (KPIs)
Monitoring	**Testing**	☐Ad-hoc testing	☐Systematic testing ☐Requirement based testing ☐Test following build	☐Integrated testing ☐Advanced systematic testing ☐Centralize management and automate test ☐test cycle every time a change is committed	☐Qualitative testing ☐Advanced automated systematic testing ☐Manage data and virtualize services for test	☐Test continuously ☐Automated recoverability and resilience testing
	Quality assurance	☐Ad-hoc quality management ☐Non-existing quality standard	☐Quality standard exists ☐Quality management exists	☐Quality metrics to measure improvement performance	☐Organized performance management	☐Continuous quality improvement (self-healing)
	Monitoring	☐Ad-hoc monitoring	☐Requirements and incidents gathering and prioritization	☐Monitor using business and end user context ☐Monitor resources consistently	☐Use enterprise issue resolution applications	☐Optimize monitoring based on KPIs
	Actions after code commits	☐Manual code quality monitoring	☐Broken build detection	☐Gated code check-in	☐Automated code quality monitoring	☐Gated commits with quality gates

		Maturity model V2				
	Capability	Level 1: Initial	Level 2: Managed	Level 3: Defined	Level 4: Measured	Level 5: Optimized
Culture	**Team Structure**	☐Isolated teams ☐Organized around skillset (dev/ops)	☐Teams structured around deliveries ☐One backlog per team	☐Teams structured around projects	☐structured around products ☐Cross-functional with knowledge overlap	☐Interdisciplinary teams and structured around KPIs ☐multi-skilled and flexible members
	Process improvement	☐All processes are manual or undefined	☐Processes and methods are defined intrateam	☐Adapting processes inter-team to make them more seamless	☐Adaption of methods and processes to all aspects of the organization	☐Continuous process improvement
	Feedback Cycle	☐Restricted feedback opportunities	☐Rapid intrateam feedback ☐Safe feedback environment	☐Rapid inter-team feedback	☐Frequent feedback on all processes	☐Continuous feedback process improvement
Automation	**General**	☐Ad-hoc delivery processes ☐Ad-hoc automation ☐Monolith structures	☐Scheduled (delivery) processes ☐First microservices exist	☐Automated (delivery) process ☐Standardized automation ☐Part of products consist of microservices	☐Frequent delivery process ☐Continuous delivery ☐Products fully consisting of microservices	☐Smart automation to maximize throughput ☐Products with optimized microservices
	Environments (test, staging, production etc.)	☐Consolidated platform and technology ☐Manually provisioned infrastructure	☐Partly automatically provisioned infrastructure ☐Organize system into modules	☐Automatically provisioned infrastructure ☐Virtualization when applicable ☐standard process across all environments	☐Manage environments through automation ☐Full component-based architecture	☐Infrastructure As Code ☐All environment configuration is externalized and versioned
	Build automation	☐Versioned code base ☐Scripted builds ☐Scheduled builds ☐Dedicated build server	☐Build automation ☐Builds are stored ☐manual tag & versioning	☐Auto triggered build ☐automated tag & versioning ☐build once and deploy anywhere	☐Build metrics gathered, made visible and considered	☐Continuous build process improvement, e.g. faster feedback
	Deployment automation	☐Manual deployment ☐some deployment scripts exist	☐Mostly standardized deploys ☐Partly automated deployment	☐Non-production deployment automation ☐basic deployment pipeline for all environments ☐automated most of the database changes	☐Production deployment automation ☐Automated database changes ☐Zero downtime deploys ☐Parallel processing with multiple build machines ☐Continuous deployment to test	☐Collaboration between teams to manage risks and reduce cycle time ☐Zero touch continuous deployments ☐Continuous deployment to production ☐Feature toggles
	DevOps tools	☐No tools used for DevOps process	☐Tools used for basic functions	☐Tools used for automatization	☐Fully integrated tools into the process	☐Optimized to the business KPIs
	Documentation & configurations	☐Not available or is outdated	☐Up-to-date documentation and configurations	☐Regular validation of the documentation and configurations	☐Documents are updated based on gathered experience and quality requirements	☐Automated generation of documentation and configurations

Maturity model V2 - continued						
	Capability	Level 1: Initial	Level 2: Managed	Level 3: Defined	Level 4: Measured	Level 5: Optimized
Collaboration & communication	**Communication & Coordination**	☐Indirect ☐Poor ☐Ad-hoc	☐Controlled ☐Facilitated ☐Actively managed	☐Standardized ☐Direct	☐Metrics for improving communication ☐Structured ☐Peer-to-peer ☐Constructive	☐Communication and coordination optimized ☐Community building Optimize communication
	Collaboration	☐ Ad-hoc collaboration	☐Active collaboration	☐Sharing pain between teams	☐Extended collaboration	☐Optimize collaboration
Measurement	**Reporting**	☐Manual reporting ☐Ad-hoc reporting ☐Only visible for the report runner	☐Scheduled quality reports ☐Tool generated reports ☐Latest reports always accessible ☐Report is visible for the whole team	☐Report history is available ☐Report is visible cross-silo	☐Report trend analysis ☐Collaboration based process measurement, which allows to identify bottlenecks and inefficacies	☐Cross silo analysis report
	Business objective focus releases	☐Manage department resources ☐Document objectives locally	☐Link objectives to release ☐Centralize requirement management	☐Plan and source strategically ☐Dashboard portfolio measures	☐Automation metrics exist to assess progress against business goals	☐Define release with business objectives ☐Measure and optimize to customer value (KPIs)
Monitoring	**Testing**	☐Ad-hoc testing	☐Systematic testing ☐Requirement based testing ☐Test following build	☐Integrated testing ☐Advanced systematic testing ☐Centralize management and automate test ☐test cycle every time a change is committed	☐Qualitative testing ☐Advanced automated systematic testing ☐Manage data and virtualize services for test	☐Test continuously ☐Automated recoverability and resilience testing
	Quality assurance	☐Ad-hoc quality management ☐Non-existing quality standard	☐Quality standard exists ☐Quality management exists	☐Quality metrics to measure improvement performance	☐Organized performance management	☐Continuous quality improvement (self-healing)
	Monitoring	☐Ad-hoc monitoring	☐Requirements and incidents gathering and prioritization	☐Monitor using business and end user context ☐Monitor resources consistently	☐Use enterprise issue resolution applications ☐Continuous monitoring of software products	☐Optimize monitoring based on KPIs
	Actions after code commits	☐Manual code quality monitoring	☐Broken build detection	☐Gated code check-in	☐Automated code quality monitoring	☐Gated commits with quality gates

		Maturity model V3				
	Capability	Level 1: Initial	Level 2: Managed	Level 3: Defined	Level 4: Measured	Level 5: Optimized
Culture	**Team Structure**	☐Isolated teams ☐Organized around skillset (dev/ops)	☐Teams structured around deliveries ☐One backlog per team	☐Teams structured around projects	☐structured around products ☐Cross-functional with knowledge overlap	☐Interdisciplinary teams and structured around KPIs ☐multi-skilled and flexible members
	Process improvement	☐All processes are manual or undefined	☐Processes and methods are defined intrateam	☐Adapting processes inter-team to make them more seamless	☐Adaption of methods and processes to all aspects of the organization	☐Continuous process improvement
	Feedback Cycle	☐Restricted feedback opportunities	☐Rapid intrateam feedback ☐Safe feedback environment	☐Rapid inter-team feedback	☐Frequent feedback on all processes	☐Continuous feedback process improvement
Automation	**General**	☐Ad-hoc delivery processes ☐Ad-hoc automation ☐Monolith structures	☐Scheduled (delivery) processes ☐First microservices exist	☐Automated (delivery) process ☐Standardized automation ☐Part of products consist of microservices	☐Frequent delivery process ☐Continuous delivery ☐Products fully consisting of microservices	☐Smart automation to maximize throughput ☐Products with optimized microservices
	Environments (test, staging, production etc.)	☐Consolidated platform and technology ☐Manually provisioned infrastructure	☐Partly automatically provisioned infrastructure ☐Organize system into modules	☐Automatically provisioned infrastructure ☐Virtualization when applicable ☐standard process across all environments	☐Manage environments through automation ☐Full component-based architecture	☐Infrastructure As Code ☐All environment configuration is externalized and versioned
	Build automation	☐Versioned code base ☐Scripted builds ☐Scheduled builds ☐Dedicated build server	☐Build automation ☐Builds are stored ☐manual tag & versioning	☐Auto triggered build ☐automated tag & versioning ☐build once and deploy anywhere	☐Build metrics gathered, made visible and considered	☐Continuous build process improvement, e.g. faster feedback
	Deployment automation	☐Manual deployment ☐some deployment scripts exist	☐Mostly standardized deploys ☐Partly automated deployment	☐Non-production deployment automation ☐basic deployment pipeline for all environments ☐automated most of the database changes	☐Production deployment automation ☐Automated database changes ☐Zero downtime deploys ☐Parallel processing with multiple build machines ☐Continuous deployment to test	☐Collaboration between teams to manage risks and reduce cycle time ☐Zero touch continuous deployments ☐Continuous deployment to production ☐Feature toggles
	DevOps tools	☐No tools used for DevOps process	☐Tools used for basic functions	☐Tools used for automatization	☐Fully integrated tools into the process	☐Optimized to the business KPIs
	Documentation & configurations	☐Not available or is outdated	☐Up-to-date documentation and configurations	☐Regular validation of the documentation and configurations	☐Documents are updated based on gathered experience and quality requirements	☐Automated generation of documentation and configurations

Maturity model V3 - continued						
	Capability	Level 1: Initial	Level 2: Managed	Level 3: Defined	Level 4: Measured	Level 5: Optimized
Collaboration & communication	**Communication & Coordination**	☐Indirect and poor communication ☐Ad-hoc collaboration	☐Controlled and facilitated communication ☐Actively managed communication	☐Standardized and direct communication	☐Metrics for improving communication ☐Structured, constructive and Peer-to-peer communication	☐optimized communication and coordination ☐Community building Optimize communication
	Collaboration	☐ Ad-hoc collaboration	☐Active collaboration	☐Sharing pain between teams	☐Extended collaboration	☐Optimized collaboration
Measurement	**Reporting**	☐Manual reporting ☐Ad-hoc reporting ☐Report visibility: report runner	☐Scheduled quality reports ☐Tool generated reports ☐Latest reports always accessible ☐Report visibility: whole team	☐Report history is available ☐Report visibility: cross-silo	☐Report trend analysis ☐Collaboration based process measurement, which allows to identify bottlenecks and inefficacies	☐Cross silo analysis report
	Requirement management	☐Manage department resources ☐Document objectives locally	☐Link objectives to releases ☐Centralize requirement management	☐Plan and source strategically ☐Dashboard portfolio measures	☐Automation metrics exist to assess progress against business goals	☐Define release with business objectives ☐Measure and optimize to customer value (KPIs)
Monitoring	**Testing**	☐Ad-hoc testing	☐Systematic testing ☐Requirement based testing ☐Test following build	☐Integrated testing ☐Advanced systematic testing ☐Centralized management of tests ☐test cycle every time a change is committed	☐Qualitative testing ☐Advanced automated systematic testing ☐Manage data and virtualize services for test	☐Test continuously ☐Resilience testing
	Quality assurance	☐Ad-hoc quality management ☐Non-existing quality standard	☐Quality standard exists ☐Quality management exists	☐Quality metrics to measure improvement performance	☐Organized performance management	☐Continuous quality improvement
	Monitoring	☐Ad-hoc monitoring	☐Requirements and incidents gathering and prioritization	☐Monitor using business and end user context ☐Monitor resources consistently	☐Continuous monitoring of software products	☐Optimize monitoring based on KPIs
	Actions after code commits	☐Manual code quality monitoring	☐Broken build detection	☐Gated code check-in	☐Fully automated code quality monitoring	☐Gated commits with quality gates
	Incident handling	☐Ad-hoc incident handling	☐Structured incident handling	☐Quick recovery on critical incidents	☐Automated recoverability	☐Predictive maintenance to prevent incidents

EXPERT INTERVIEW PROTOCOL – DATA ACQUISITION

1 Introduction Interview

My name is Jeroen Radstaak and I am currently completing the master Business Information Technology at the University of Twente, The Netherlands. Currently, I am doing my thesis project at Hitachi Consulting Vietnam. My thesis covers the topic of DevOps capabilities and maturity model. The aim is to make a comprehensive model covering aspects of organizations that should be considered to improve their DevOps approach.

The interview should not take any longer than 60 minutes. The goal is to use your expertise in this field to get an insight into DevOps practices and how these relate to what I have found in literature. Questions will be asked regarding your organization and capabilities needed for DevOps. The capabilities and practices elicited through this interview will be input for the DevOps maturity model.

2 Main part of the interview

2.1 General questions

Please answer the following questions, so I can get a complete picture of who you are and what you do:

1) What is your role in the organization?
2) What department are you in?
3) How long have you been working in similar roles (including other organizations)?
4) How long have you been working for this organization?

2.2 DevOps and Maturity Model definitions

No formal definition of DevOps has been found in the literature. However, the term has been around for a decade. This brings me to the question

1) What does the term “DevOps" mean to you?
2) What departments do you consider to be involved in the software development process? (any outside of Development and operations)
3) What does a maturity model mean to you? Can you think of an example?

2.3 DevOps automation

DevOps is known to be enabling automation. Benefits attributed to the automation are reduced cycle times and enabling continuous deployment of high-quality software. Continuous integration is an important keyword in the DevOps automation.

1) How do you describe continuous integration?
2) What would you consider to be the advantages of the automation that is associated with DevOps?

3) What processes, that are part of the software development cycle, are currently automated?

4) How are these processes automated? What automation practices have been implemented?

5) What future automation practices will be implemented to automate more processes

6) What would be your long-term goal regarding a realistic automation setup for a project that uses DevOps?

2.4 DevOps Monitoring

To be able to measure processes they first must be monitored. This can than result in improving the processes.

1) What type of processes are monitored? Business/technical?

2) What is the process for testing? Automated? Periodical?

3) What practices have been implemented by the organization to monitor processes that are of interest? (QA, testing, code commit actions)

4) What future practices would you like to see implemented for monitoring?

5) Is collaboration between teams monitored?

2.5 DevOps Measurement

For DevOps it is considered important to measure processes against certain metrics. This enables continuous improvement.

1) What processes should be measured according to you?

2) Which processes are currently measured in your organization?

3) Are the customers considered in the decision on what to measure?

4) Are the results of the measurements reported? And how, to whom?

5) What would you, considering your current practices, want to implement in the future for measurement?

2.6 DevOps collaboration & communication

DevOps focuses on a fostering culture of collaboration and communication.

1) Is knowledge and experience shared between teams at your organization?

2) Is there frequent communication between development and operations? (how frequent?)

3) What are issues that arise in the communication? Can you give some examples of how this happened in the past?

4) Have any practices been implemented to prevent issues in communication?

5) What future practices should be implemented in your opinion?

2.7 DevOps culture
Last, but not the least, we will discuss the culture that is needed to foster a DevOps environment.

1) Is the culture in a DevOps environment important to it succeeding?
2) What does a culture of trust and respect mean to you?
3) How would you describe the culture in your organization? (Team structure, feedback etc.)
4) What practices have been implemented to improve trust and respect in the organization?
5) What would be the goal for your company regarding culture changes, considering your current practices?

3.0 Completing the maturity model
Based on everything that we have discussed, I think it is interesting to complete the maturity model that I will show you now. (figure D.1)

1) Why do you think that this ∗ capability∗ fits best in this ∗ category∗ ? (*Ask repetitively while completing the model*)

An example of a maturity model is shown to the participants to give an idea about what their input will be used for and to help explain what a maturity model is.

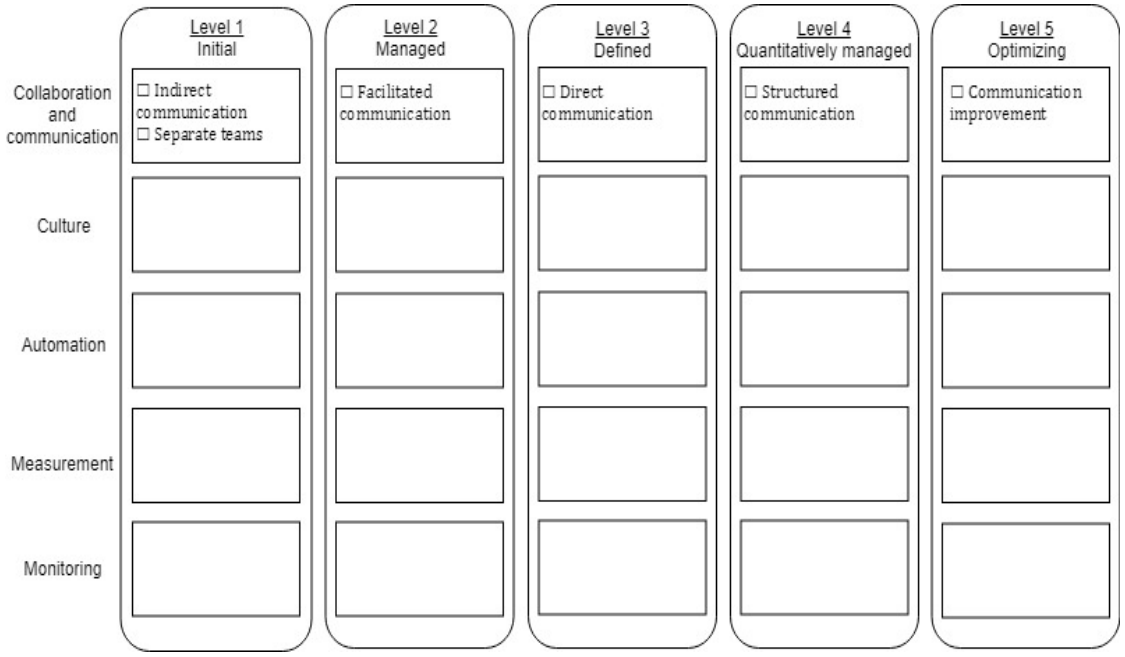

Figure D.1: Example of DevOps maturity model with first row partly filled in

4.0 Ending interview
Thank you for your time and the participation in this interview. This was the information that I needed for my thesis. After the thesis is completed, I will send it to you.

E

CASE STUDY - RESEARCH PLAN VALIDITY

To ensure the validity of the case study the test described by Yin will be executed. These tests are to establish the construct validity, internal validity, external validity and reliability [57]. Each of these will be discussed in the following sections.

E.1 CONSTRUCT VALIDITY

Construct validity is the identification of correct operational measures for the concepts that are being studied. Various tactics can be adopted to handle the construct validity in a case study, such as using multiple sources of evidence, establishing a chain of evidence during data collection and having key informants review the draft report [57].

The construct in this case study is the measurement of the DevOps maturity level of software developing organizations. This also serves the purpose of evaluating the maturity model and its capabilities in practice to provide an organization with advice on improving the maturity.

Case studies are generally limited in construct validity, as case study research is often open to interpretation which leads to subjectivity. In this case study it was attempted to limit subjectivity by collecting data from multiple sources, combining literature and expert interviews to create the maturity model. Also, many of capabilities in the DevOps maturity model can be objectively measured. However, some capabilities must be interpreted and can therefore create a subjective bias.

The chain of evidence is established by using a case study protocol, a maintained database of case study data and reporting of the case study as was done in this report. The final aspect is to have the draft case study report reviewed by key people, which in this case is the exam committee. As all three have been considered it can be said that construct validity is met.

E.2 INTERNAL VALIDITY

The internal validity is to seek a causal relationship between x and y. Yin states that this is for explanatory and casual studies only and not for descriptive or exploratory studies [57]. The case study performed in this research is an exploratory case study. Thus, the internal validity does not need to be tested for this case study.

E.3 EXTERNAL VALIDITY

The external validity of a case study is described by Yin as “showing whether and how a case study’s findings can be generalized" [57]. This can be achieved

by using a replication logic and by doing multiple case studies, thereby ensuring that the findings can be applied to other situations. Regarding this research, the case study was performed at multiple organizations as well as the interview data and literature sources used to establish the completed maturity model.

However, even though measures have been taken to improve the generalization of this research it is impossible to be completely certain of external validity. The amount of case studies is very small in comparison to the usage of DevOps in organizations around the world. Especially considering that all the organizations are in Vietnam, which could lead to a cultural bias. This results in a limited generalizability of the research worldwide, but possibly a high generalizability in Vietnam or (South)East-Asia.

E.4 RELIABILITY

The objective for the reliability test is that if another researcher follows the same procedures as described by the previous researcher that this researcher will arrive at the same findings and conclusions [57]. This was achieved by creating and following a case study protocol and by developing a case study database in this research as detailed as possible. It is not possible to do the exact same case study as the organizations and participants have been promised anonymity. However, it would be expected that another researcher will reach the same findings and conclusions as set out in this research.

E.5 RESEARCH PLAN CONCLUSIONS

In the previous sections the case study performed in this research have been tested for construct validity, internal validity, external validity and reliability. None of the tests were flawless regarding this case study. However, no major threats were identified.