Master Thesis
for the study programme MSc. Business Information Technology

DEVOPS UNDER CONTROL

OLIVIA H. PLANT

Development of a framework for achieving internal control and effectively managing risks in a DevOps environment

March 2019

UNIVERSITY OF TWENTE

Olivia H. Plant: *DevOps under control,* Development of a framework for achieving internal control and effectively managing risks in a DevOps environment
Master Thesis, *University of Twente,* March 2019

AUTHOR:
Olivia H. Plant, MSc candidate

Study Programme:	MSc Business Information Technology
Track:	IT Management & Innovation
E-Mail:	o.h.plant@alumnus.utwente.nl

GRADUATION COMMITTEE:
Dr. Klaas Sikkel

Faculty:	Electrical Engineering, Mathematics and Computer Science
Department:	Services and Cybersecurity
Email:	k.sikkel@utwente.nl

Prof. Dr. Jos van Hillegersberg

Faculty:	Behavioural, Management and Social Sciences
Department:	Industrial Engineering and Business Information Systems
Email:	j.vanhillegersberg@utwente.nl

Frank van Praat, MA MSc RE

Company:	KPMG Nederland
Department:	IT Assurance & Advisory Noord
Position:	Senior Manager
E-Mail:	vanpraat.frank@kpmg.nl

MANAGEMENT SUMMARY

Although multiple definitions of the DevOps concept exist, DevOps is generally considered to be an Agile software development approach with the goal of combining development and operations and emphasizing frequent and fast software deployment. Four main aspects of DevOps are *collaboration*, *automation*, *measurement* and *monitoring*. While the DevOps approach offers great benefits, many companies are struggling with the implementation of DevOps and with maintaining control of their processes due to the required autonomy of the DevOps teams and the high degree of automation. At the same time, they struggle with demonstrating this control towards external auditing parties. This study therefore seeks to identify which types of risks companies using DevOps are generally exposed to and to develop a framework that helps companies control their processes and manage risks without hindering the speed and efficiency of the DevOps approach substantially.

The literature review suggests that many risk management controls concerning access management, change management, compliance and security can be automated. However, research on DevOps is still scarce and specific risks applicable to DevOps are hardly mentioned. Furthermore, we conducted case studies in nine companies using DevOps which show that manners of implementing DevOps differ widely and that many companies in practice use a combination of traditional and automated controls to manage their DevOps environment. This study also shows that soft aspects such as organizational culture, communication and team responsibility are of integral importance for effectively mitigating risks in DevOps.

Risks associated with DevOps can be grouped into five categories which are *transitional, organizational, project, team* and *product* risks. It is further argued that there is no best way to implement DevOps and that the DevOps concept rather needs to be tailored to the needs of the company in question. Two main factors that influence companies in their decision how to manage their processes are the *DevOps maturity* and *risk appetite*. Based on these factors, a framework is developed that suggests four strategies with suitable controls to manage risks in DevOps.

The findings of this study implicate that companies first have to find a way to establish a solid DevOps culture before relying on automation practices. Likewise, auditors will have to find a way to assess these so-called "soft controls" in order to reliably give assurance on internal control. This thesis presents some first suggestions on how this can be done.

This study has both scientific and practical value:

- *scientific*: The study is one of the first of its kind and contributes to the scarce field of research about risk management in DevOps.
- *practical*: We provide guidelines for companies on how to integrate risk management practices into their DevOps processes.
- *practical*: We provide insights for auditors who want to provide assurance on internal control on how to audit DevOps processes.

The main research methods applied were a structured, multivocal literature review and multiple case studies that draw from semi-structured interviews as main input. The framework has been validated with four experts in the fields IT Risk, IT Audit and DevOps and with six of the case study participants.

The findings are mainly limited by the fact that some case studies draw from only a single interview as source of information and that the validation techniques applied were artificial instead of naturalistic. Directions for future research include implementation and auditing of soft controls, assessing DevOps maturity and general success factors for DevOps transitions.

PREFACE

After five and a half years, my life as a student at the University of Twente comes to an end. It has been a time full of adventures and personal growth, starting with a Bachelor in International Business Administration and finally finding my passion for IT Management and moving on to pursuing the MSc programme in Business Information Technology. This thesis is the result of nine months work and I am quite happy with how it turned out. Many people have contributed to this research over the past few months and I would like to thank some of them in the following:

I would firstly like to thank my supervisors Klaas Sikkel and Jos van Hillegersberg. I could not have wished for a better combination of supervisors. Both of them have always made time for me, supported all of my ideas and given me valuable feedback. Thank you for the interesting meetings and the hours you must have spent on reading my drafts.

I would also like to thank my KPMG colleagues for their advice and interest in my thesis, especially my company supervisor Frank van Praat who always made time for me despite his busy schedule and my fellow internship companions who kept me motivated throughout both coffee breaks as well as stressful periods. I would also like to thank Henk Hendriks for pointing me towards this topic in the first place; this thesis has sparked my enthusiasm about Agile and DevOps even further and I thoroughly enjoyed working on it.

I am also grateful to all people who made time to participate in the case studies and/or give me feedback on the results. Their insights and contributions mark the core of this thesis and this research would have been much less interesting without their participation.

Most importantly, I would like to thank my friends and family, especially my parents and my boyfriend, for always supporting me throughout my studies. This thesis would not have been possible without you.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

AWS	Amazon Web Services
CAB	Change Advisory Board
CD	Continuous Deployment
CI	Continuous Integration
COBIT	Control Objectives for Information and related Technology
COSO	Committee of Sponsoring Organizations of the Treadway Commission
DTAP	Development-Testing-Acceptance-Production
ERM	Enterprise risk management
GITC	general IT controls
IaC	Infrastructure as Code
SAFe	Scaled Agile Framework
SOx	Sarbanes–Oxley Act

1 INTRODUCTION

DevOps is often used as an umbrella term to describe software development approaches with the aim of increasing the pace of software development processes and improving software quality [12]. Important practices often found in DevOps teams are the shared responsibility for software development and operations and sometimes at least partly automated software delivery pipelines and infrastructure.

A detailed description of the DevOps phenomenon is given in Section 2.1

Formerly known for its use in more technologically advanced companies such as Netflix, Etsy and Spotify, DevOps has also become interesting for more traditional companies [24] and is nowadays continuously gaining popularity [39, 10]. While many companies are enthusiastic about the opportunities that DevOps offers and are keen to implement it, they are struggling to maintain control of their processes due to the high degree of automation as well as the required autonomy of the DevOps teams and decentralized decision making structures. It is therefore beneficial to adopt a more tailored and risk-management based approach when designing the DevOps processes for a company. A second struggle for companies as well as for their auditors is to demonstrate this control in IT audits. Traditional control frameworks that stress aspects such as change control, access management and security are no longer compatible with DevOps and Agile ways of working and need to be adjusted.

Despite the obvious need for more rigorous investigations of these problems, academic research is only recently picking up on the DevOps trend with publications having increased significantly over the past three years. However, much of the available literature is still concerned with defining what DevOps is in the first places or focuses only on the technical aspects of automation. The research at hand aims at setting a first step towards more structured risk management and process design in DevOps with the goal to increase internal control while remaining as agile as possible.

1.1 THESIS STRUCTURE

This thesis is structured as follows:

- Chapter 2 gives a detailed overview of the DevOps concept and attempted definitions by scholars as well as a short description of risk management and internal control terminology.
- Chapter 3 describes the design of this research and the steps to be undertaken during the course of its execution.
- Chapter 4 summarizes the academic literature available in the context of DevOps and risk management.
- Chapter 5 explains the empirical research methodologies in detail.

- Chapter 6 summarizes the empirical case study results.
- Chapter 7 synthesizes the results of the literature and empirical studies and introduces the final risk management framework.
- Chapter 8 summarizes the validation of the initial draft framework and accounts for the adjustments that were made to this.
- Chapter 9 discusses the implications of the results for scholars and practitioners and evaluates their validity, reliability and limitations.
- Chapter 10 sums up and concludes the thesis.

2 BACKGROUND

2.1 WHAT IS DEVOPS?

DevOps is a combination of the words *development* and *operations* and was first used during a presentation by Patrick Debois and Andrew Clay Shafers at the 2008 Agile Conference [27]. The central philosophy of DevOps which scholars and practitioners agree on is that DevOps aims to bridge the gap between development and operations by assigning DevOps teams shared responsibility for both processes [26, 44]. DevOps has been referred to as many different things among which a movement, a philosophy, a (development) practice, a mindset or a culture. Furthermore, there are tensions as to whether DevOps is mainly about culture or is more of a technical solution [26]. Lichtenberger [24] explicitly warns his readers that DevOps is no framework or standard that could be looked up in a codified book, but is rather a movement with the goal of becoming "better" and "faster". Literature reviews have also shown that there is no uniform definition of DevOps [12, 25] although various studies have defined some general patterns that DevOps processes usually share. In the following we will first name some definitions of DevOps as encountered during a literature review. We will then elaborate on some practices that are often associated with DevOps.

In order to apply to a wider context, this study purposely does not rely on one specific definition of DevOps.

Table 2.1: DevOps capabilities and enablers according to Smeds et al. [44]

Capabilities	Continuous planning
	Collaborative and continuous deployment
	Continuous integration and testing
	Continuous release and deployment
	Continuous infrastructure monitoring and optimization
	Continuous user behavior monitoring and feedback
	Service failure recovery without delay
Cultural Enablers	Shared goals, definition of success, incentives
	Shared ways of working, responsibility, collective ownership
	Shared values, respect and trust
	Constant, effortless communication
	Continuous experimentation and learning
Technological Enablers	Build automation
	Test automation
	Deployment automation
	Monitoring automation
	Recovery automation
	Infrastructure automation
	Configuration management for code and infrastructure

Lwakatare et al. [25] defined *collaboration*, *automation*, *measurement* and *monitoring* as the four main dimensions of DevOps. In another paper they added a fifth dimension called *culture* [26]. Similarly, Smeds et al. [44] defined DevOps as a set of capabilities, cultural enablers and technological enablers which are shown in Table 2.1. Jabbari et al. [20] found that *"DevOps is a development methodology aimed at bridging the gap between Development and Operations, emphasizing communication and collaboration, continuous integration, quality assurance and delivery with automated deployment utilizing a set of development practices"*. According to Nielsen et al. [33], DevOps incorporates three main principles which are working according to *agile principles* with continuous and frequent software delivery, *collaboration* with a culture based on trust, respect and communication and *integration* of practices and tools. The software delivery process is divided into the four stages: *plan & measure*, *develop & test*, *release & deploy*, as well as *monitor & optimize*. A literature review by Erich et al. [12] showed that research papers about DevOps emphasized the aspects *culture of collaboration*, *automation*, *measurement*, *sharing*, *services*, *quality assurance* and *governance*.

A model that is commonly used by DevOps practitioners is the competence model from the DevOps Agile Skills Association (DASA) [8] which emphasizes twelve skill and knowledge areas that should be present in DevOps teams. These areas are summarized in Table 2.2. In the following, the four original dimensions of Lwakatare et al. are used to summarize the most common DevOps practices.

2.1.1 *Collaboration*

DevOps teams have shared goals, shared incentives and shared responsibilities for development and operations [20]. Collaboration is enforced through information sharing and broadening of team members' skillsets [25]. Due to this new way of working, DevOps requires a complete shift in culture. DevOps culture is based on trust, respect and communication [33] and is one of the most difficult parts to implement for companies when moving towards DevOps [2].

Furthermore, DevOps is considered by many authors to be an extension of agile software development that aims to apply the agile principles not only to the development but also the operation of software [20, 27]. Some authors see

Table 2.2: DASA DevOps competence areas [8]

SKILL AREAS	KNOWLEDGE AREAS
Courage	Business value optimization
Teambuilding	Business analysis
DevOps leadership	Architecture and design
Continuous improvement	Programming
	Continuous delivery
	Test specification
	Infrastructure engineering
	Security, risk and compliance

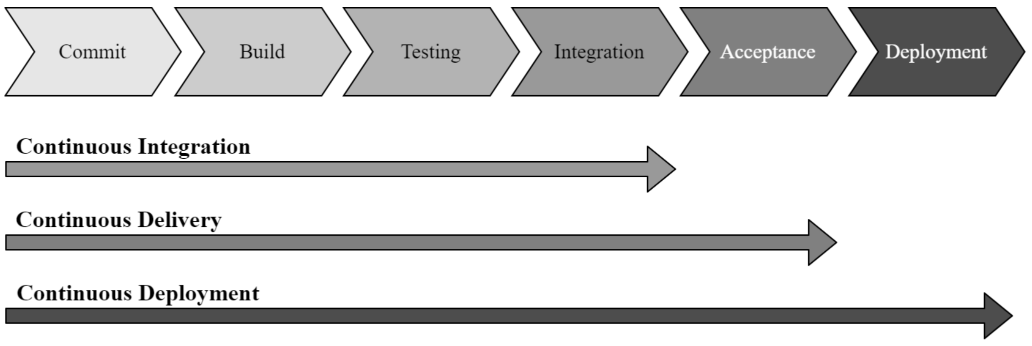


Figure 2.1: Typical automated activities included in CI and CD practices

agile as an enabler for DevOps while only few authors see agile as a separate development methodology with similarities to DevOps [20]. Both focus on rapid and incremental releases, gathering feedback quickly and correcting problems [27].

2.1.2 *Automation*

According to Lwakatare et al. [25], increased automation of testing and deployment processes is necessary to keep up with the increased pace of agile software development. Three terms that are repeatedly mentioned in combination with DevOps but are often used interchangeably are Continuous Integration, Continuous Delivery and Continuous Deployment [46]. *Continuous Integration (CI)* is a development practice where team members integrate their work [46] by constantly merging working copies to a shared main branch [22]. Changes in code are directly tested and merged in order to continuously validate the code and detect problems as early as possible. As soon as a developer commits a change, the system detects this automatically and triggers a build, conducts automated tests and posts the build to a repository [52]. *Continuous Delivery* builds on this concept by additionally preparing the software for release. In order to do so, automated acceptance tests are conducted and the code is deployed to a staging environment. The software can then be deployed with a single manual click on a button [36]. Finally, *Continuous Deployment (CD)* extends the two principles by also conducting an automated release process following the extensive testing [46]. In this case, no human interaction is needed in order to deploy a change once a piece of code is checked in by a developer and passes all automated tests. Figure 2.1 visualizes the described activities and differences between these three principles. While many companies organize their delivery process and corresponding environments according to the Development-Testing-Acceptance-Production (DTAP) approach, the exact order in which the described activities are conducted may vary per company. In order to build these automated toolchains, developers can use automation software like Jenkins which connects and triggers the necessary applications.

DevOps is generally agreed by scholars to be based on Lean thinking and aims to make processes more efficient and effective throughout the entire IT value stream [27]. In order to implement Continuous Deployment efficiently, parts of the development and deployment chain that do not add value should therefore be eliminated and features that are ready for delivery can be released immediately [22].

Another important principle that is frequently used within DevOps is known as *Infrastructure as Code (IaC)* and is often used for configuration management of servers that will run the applications. The desired state of infrastructure and configurations is defined in a domain-specific language [7]. This configuration information is then stored in source code repositories. Tools such as such as Chef, Puppet, Salt or Ansible allow developers to treat these configurations as code which can be versioned and tested and ultimately rolled out by ensuring that all systems have the defined configurations [40, 43]. This can for example be used to ensure that the acceptance environment in which code is tested is the same as the actual production environment to which the changes will be deployed once they pass the tests successfully.

A similar technique that is gaining popularity in DevOps are containers like Docker. These containers are quick to set up and provide a separate environment for applications to be tested and developed in. They are often used to create virtual development, test and production environments in DevOps [26]. Docker containers are launched from images that contain information about their content such as applications and processes to be run once the container is launched. They can be distributed via registries which makes Dockers very portable. Different to virtual machines, Docker runs on top of the host operating system and does not require installment of another operating system. They are therefore very resource efficient [41]. However, configurations in Docker containers cannot be changed since containers cannot be updated. Updated software or configuration therefore requires a new image build [40].

2.1.3 *Monitoring*

Monitoring allows for fast detection and correction of problems which is central to DevOps. Systems and the underlying infrastructure should therefore be continuously monitored by operations personnel. Furthermore, continuous monitoring allows for appropriate assignment of resources [25]. Monitoring is conducted by implementing automated monitoring tools and logs. However, it can be difficult for development personnel to search the large amount of available logs to detect anomalies if the systems are not designed to show errors automatically. Furthermore, Continuous Deployment somewhat challenges monitoring due to its focus on speed and effectiveness. DevOps addresses these problems by emphasizing collaboration between development and operations personnel so systems are designed to expose relevant information quickly [25].

2.1.4 *Measurement*

Quality assurance is mentioned as an important part of DevOps by multiple authors [20, 25]. Integrating measurement into the DevOps pipeline ensures that performance of development and quality assurance is based on quantitative data. Measurement should be based on real time performance and usage data [25]. The metrics to be measured should always focus on business value of the operations and production data should drive decisions, improvements and changes to the system [26].

2.2 RISK MANAGEMENT AND INTERNAL CONTROL

The international risk management standard ISO31000 defines risk management as a set of principles, frameworks and processes for managing risk [45]. However, the Committee of Sponsoring Organizations of the Treadway Commission (COSO) [4] has shifted the focus to a more holistic view of risk management with the establishment of its Internal control and Enterprise risk management (ERM) frameworks. In these frameworks, COSO advocates for the implementation of appropriate risk-based controls throughout the enterprise to ensure the achievement of organizational objectives. This infers that risk management impacts organizational management as a whole instead of only applying to risk management processes [45]. Risk management therefore is also no longer just a function focused on financial and accounting risks but on management control throughout the whole enterprise [45]. Furthermore, internal control is an integral part of ERM according to COSO [4].

2.2.1 *The Sarbanes–Oxley Act*

Corporate scandals in the early 2000s have lead to the establishment of the Sarbanes–Oxley Act (SOx) which requires companies to regularly report on their internal control structure and procedures concerning financial reporting together with independent auditors [1, 49]. In order to do so, companies often adopt frameworks such as COSO which help them to structure their control processes. Although COSO is one of the most popular frameworks used for SOx compliance, one of its limitations is that it does not explicitly name any control concepts [42]. Another framework which is often used and does provide specific controls and processes is the Control Objectives for Information and related Technology (COBIT) framework [42]. Rubino et al. [42] advocate that COBIT can be a useful internal control framework for companies which overcomes some of COSO's limitations.

2.2.2 *IT audit and controls*

The audit committee oversees the financial performance of the enterprise and ensures the reliability of its financial reporting [49]. Internal control is therefore important in IT Audits which are conducted as part of the financial statement audits but also for achieving (security) certifications. According to Gantz [15], IT Audit can help organizations ensure that assets are governed effectively i.e. they operate as intended and work in a way that complies with applicable regulations and standards.

IT controls are items that are tested and analyzed during an IT audit and thus form the substance of auditing [15]. There are various approaches to categorizing IT controls: They can be preventive, detective or corrective towards the risks they address and can be of administrative, technical or physical nature [15]. Administrative controls concern policies, procedures or plans to ensure the

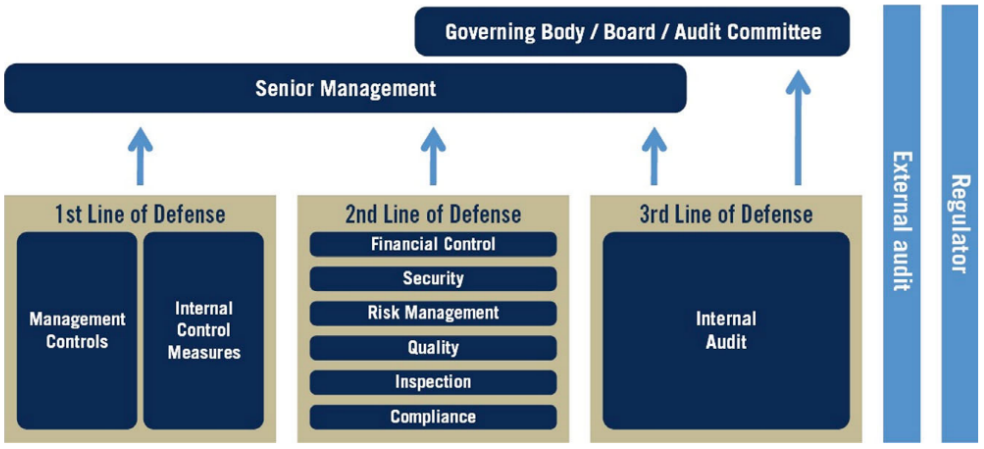


Figure 2.2: Three lines of defence model as illustrated by Davies and Zhivitskaya [5]

integrity of the organizations operations and assets while technical controls are designed to achieve the organizations control objectives. Physical controls concern the access to facilities and assets. Auditors also distinguish between general IT controls (GITC) and application controls [19]. Application controls are integrated into applications which support the financial control objectives such as financial applications. Application controls for example include *completeness* and *accuracy* of the working of an application. The GITC are integrated into IT processes which ensure a reliable operating environment and support the application controls. Examples for these controls are *access to programs and data* and *changes to programs* [19].

2.2.3 *Three lines of defence model*

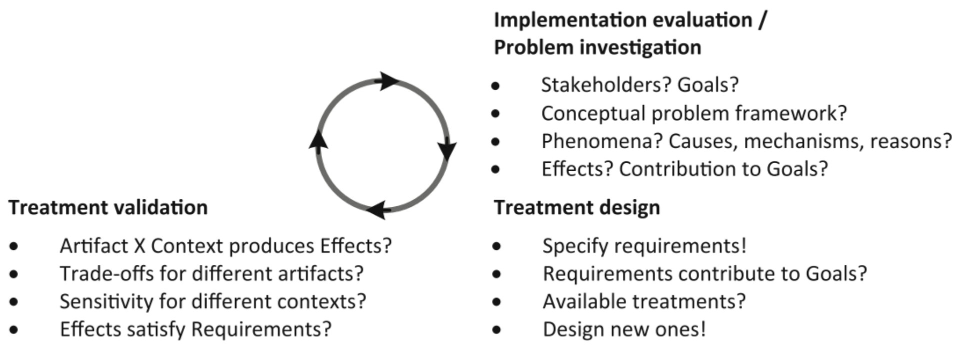
Many companies arrange their corporate governance according to the "Three lines of defence" model. The model is particularly popular in financial institutions but also used in other sectors [47]. The exact origins of it are unknown and there are multiple subtly different version of it in which the boundaries between the three lines differ slightly [5]. The model divides organizational risk management activities into three "lines of defence". The first line of defence is designed to reduce operational risk in day-to-day activities. It contains management and internal controls and is performed by the individual employees and their superiors. The intention of this first line is to capture risks early and prevent them from happening [47]. The risk ownership is maintained by the business. The second line contains overseeing and supporting functions. Most importantly, this includes the central risk management organization but also supporting functions like compliance, legal and HR [5]. This line sets the company-wide rules and policies and provides risk owners from the first line with information about risks across the organization [47]. The third line of defence is the internal audit which provides assurance on the effectiveness of first two lines. As shown in Figure 2.2, the lines of defence are overseen by the senior management and governing bodies and are assessed by external auditors.

Section 2.1 of this chapter was adapted from a previously issued report by the same author [37]

3 RESEARCH DESIGN

This research project contains a descriptive research part and design research. The descriptive research is conducted in form of literature reviews and case studies and aims to answer knowledge questions whose understanding will ultimately aid in designing an effective risk management framework. Throughout the whole project we will follow the design science methodology by Wieringa [54] which is aimed at conducting design research but also gives room for answering supporting knowledge questions. According to Wieringa, the goal of a design project is to (re)design an artifact so that it better contributes to the achievement of a goal. The design cycle describes the process of such a design research project and encompasses the phases *problem investigation*, *treatment design* and *treatment validation*. The design cycle is part of a bigger engineering cycle which also incorporates the steps of *treatment implementation* and *implementation evaluation*. Depending on the outcome of the treatment validation phase, the cycle potentially has to be iterated several times until the designed artifact produces the desired effects. The design cycle and the question corresponding to each phase are shown in Figure 3.1. Question marks represent knowledge questions while exclamation marks indicate design problems.

For a description and motivation of the research methodologies refer to Section 4.1 and Chapter 5


Implementation evaluation / Problem investigation
- Stakeholders? Goals?
- Conceptual problem framework?
- Phenomena? Causes, mechanisms, reasons?
- Effects? Contribution to Goals?

Treatment design
- Specify requirements!
- Requirements contribute to Goals?
- Available treatments?
- Design new ones!

Treatment validation
- Artifact X Context produces Effects?
- Trade-offs for different artifacts?
- Sensitivity for different contexts?
- Effects satisfy Requirements?

Figure 3.1: Design cycle adapted from Wieringa [54]

As demonstrated by Wieringa himself, the design cycle follows essentially the same steps as the design science research methodology by Peffers et al. [34]. In this research it was decided to use the methodology by Wieringa because his approach encompasses a complete conceptual framework including classification of research problems, research questions and research methods that can be applied in every step of the design cycle.

3.1 RESEARCH OBJECTIVE AND QUESTIONS

Using the template for defining design problems (also known as technical research problems) by Wieringa in Figure 3.2, the objective of this research can be formulated as to *improve risk management in DevOps by designing a framework* *that satisfies agility requirements in order to help companies demonstrate control over their processes and create valid audit trails.*

Improve <a problem context>
by <(re)designing an artifact>
that satisfies <some requirements>
in order to <help stakeholders achieve some goals>

Figure 3.2: Template for design problems [54]

The main research question results from this design research objective and is formulated as follows:

What is a suitable framework that allows companies to mitigate risks and exercise control over their DevOps environment while remaining agile?

In order to design an effective risk management framework and the desired interaction with the problem context, two descriptive knowledge questions have to be answered first and suitable implementation strategies have to be designed. Firstly, the risks that companies are dealing with have to be identified. Understanding these will aid in designing effective response strategies and controls. However, the risks are expected to differ per company and their respective environment. This research therefore aims at identifying risk categories and will investigate whether these categories differ per context. The first research sub-question is therefore defined as follows:

1. What types of risks are companies using DevOps exposed to?

The second sub-question aims at identifying specific practices that ensure an adequate management of risks in DevOps. This includes IT controls as well as existing strategies and frameworks. As indicated in the main research question, the goal of these practices is to mitigate the risks identified in research sub-question one while hindering the efficiency and agility of DevOps as little as possible. It is therefore necessary to not only consider whether these controls sufficiently mitigate risks but also to assess the impact they have on the efficiency of the process.

2. Which practices exist that can be incorporated into a DevOps process to demonstrate control and ensure the creation of valid audit trails?

Lastly, a suitable strategy for implementing these practices and addressing the identified risks has to be designed based on the outcome of the knowledge questions above.

3. Which strategy should companies drive in order to identify risks and implement suitable controls?

Research sub-question one aims at establishing a better understanding of the problem context while questions two and three are designed to illustrate the content of the risk management framework. The first two sub-questions are descriptive knowledge questions and the third question is a design problem [54].

3.2 RESEARCH MODEL

Combining the research design and questions leads to the research model presented in Figure 3.3 according to the notation by Verschuren and Doorewaard [51]. Data will be gathered both through a structured literature review as well as through case studies. This type of research was selected due to the evident lack of empirical research concerning risk management in DevOps. The individual outcomes of these studies will be synthesized to a conceptual model explaining the problem context, as well as risk mitigation strategies and controls which can be implemented in DevOps. This draft framework will then be discussed with risk management, audit and DevOps experts and will be presented to the interviewees of the case studies who will evaluate the use of the model for their company. Their input will be used to create the final DevOps risk management framework and guidelines. The arrows at the bottom of the diagram indicate the phase of the design cycle that corresponds to the particular actions. The application of the design cycle to this research is also demonstrated in Table 3.1 in more detail.

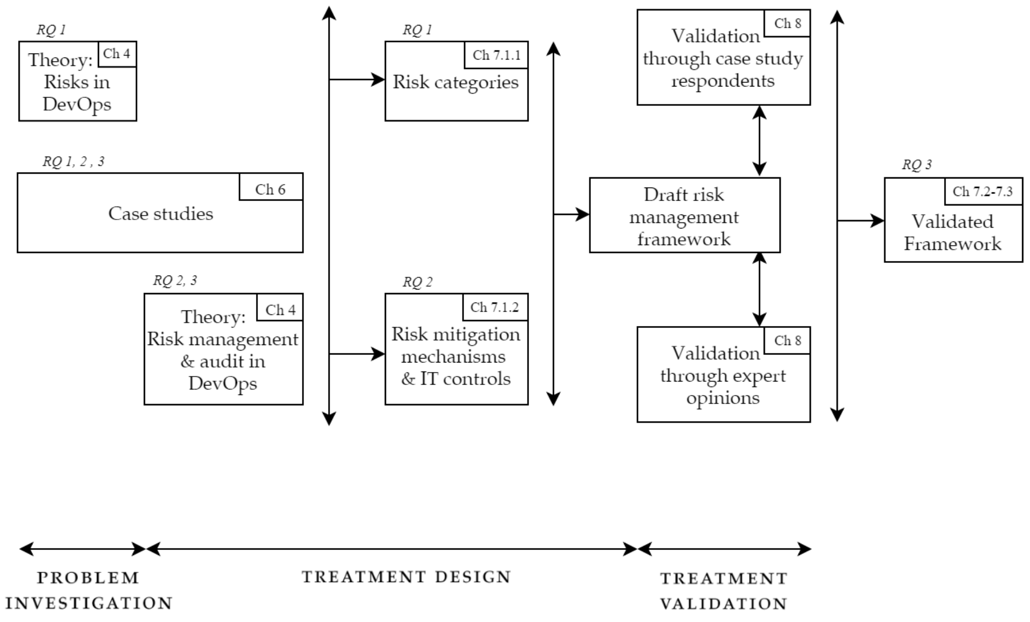


Figure 3.3: Research model

Table 3.1: Design cycle phases applied to this research

RESEARCH FOCUS	METHOD	CHAPTER
Problem investigation		
State of the art of risk management in DevOps research	Literature review	4
RQ1: What types of risks are companies using DevOps exposed to?	Literature review Case studies	4, 6
Treatment design		
RQ2: Which practices exist that can be incorporated into a DevOps process to demonstrate control and ensure the creation of valid audit trails?	Literature review Case studies	4, 6
RQ3: Which strategy should companies drive in order to identify risks and implement suitable controls?	Literature review Case studies Framework design	4, 6, 7
Validation		
Risk mitigation effectiveness	Expert opinions Case study respondents	8
Agility requirements	Expert opinions Case study respondents	8

4 LITERATURE REVIEW

4.1 LITERATURE REVIEW METHOD

In order to gain an overview of all relevant literature concerning risk management and DevOps, a structured literature review following the procedure suggested by Kitchenham [21] was conducted. An important part of structured literature reviews is the search protocol which can be found in Appendix A. This protocol provides details of search terms and inclusion and exclusion criteria in order to ensure a coherent and non-biased selection of relevant literature. The search keys were created based on an exploratory literature review. The papers were selected by first scanning the titles of all results and excluding obviously non-relevant papers. Subsequently, the abstract of papers that seemed to meet the inclusion criteria was scanned and non-relevant papers were again excluded. Lastly, the full text of the remaining papers was read before deciding which papers should be included in the review. These steps were carried out conservatively, meaning that if it was doubtful whether a paper met inclusion criteria, it was taken to the next step and was only excluded once it was certain that it would not contribute to our research. This process as well as the amount of papers left after each step of the review is shown in Figure 4.1. A more detailed overview is given in the search protocol.

In order to assure the quality of the literature, only academic databases where researched in the first place and only journal articles and conference papers where considered for inclusion in the review. However, according to Garousi et al. [16] it is important to also include so called "grey literature" (non-peer reviewed literature) in software engineering research, especially if the field of research does not provide a substantial amount of literature like it is the case with DevOps. Grey literature can provide the researcher with state-of the art concepts that might not be mentioned in academic literature and may help

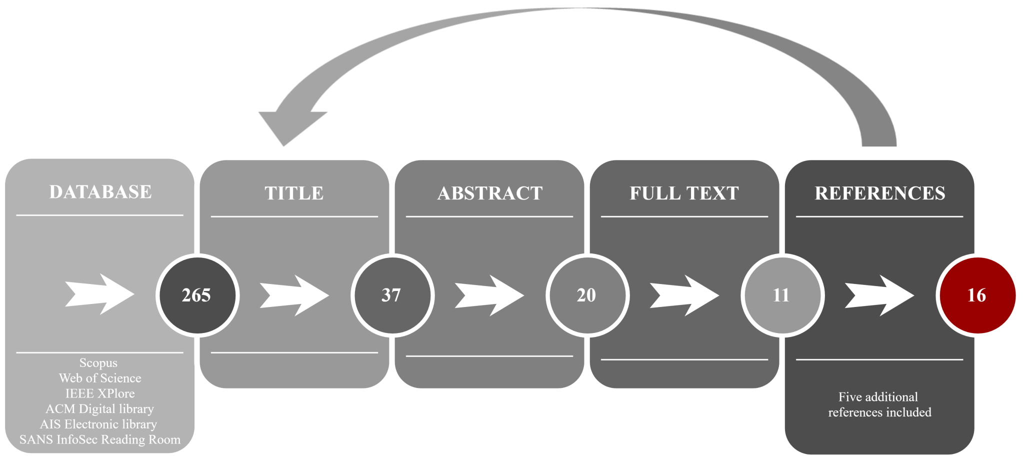


Figure 4.1: Literature review method and output

avoiding publication bias. Since there was only very little academic literature available focusing specifically on risk management in DevOps, it was therefore decided to conduct a multivocal literature review for this item following the guidelines of the aforementioned authors [17]. However, it is important to pay special attention to the quality of the papers when conducting a multivocal literature review. We therefore only included first tier grey literature with a high credibility like whitepapers and books. During the database search only literature from 2014 onwards was selected in order to gain an overview of the state of the art research. However, during the reference searches older literature was included to gain a deeper understanding of the background of the papers. This review process resulted in a list of 16 papers. Another exploratory literature search conducted for verification purposes in Google Scholar yielded no new results so the final list is deemed to be complete.

A list of papers selected for the literature review and their validation methods can be found in Appendix B.1

Of the selected papers, nine papers are conference papers and five papers are white papers. Only one journal paper and one book chapter were included. Notably, a large amount of the papers (especially white papers) are solely based on expert opinions which are in many cases the personal opinions of the authors. Furthermore, some papers were based on literature studies. Only few researchers conducted empirical research like interviews and case studies or validated their models. However, according to Kitchenham [21], software engineering research usually has little empirical evidence and scholars in this domain often have to rely on expert opinions.

In order to structure the literature review and to put the available information in the context of risk management, the literature review follows the categories of the original COSO Enterprise Risk Management Framework [4] as shown in Figure 4.2. The ERM framework was chosen because it is generally considered to be the most high-level risk management framework available and spans risk management categories throughout the whole enterprise. Although DevOps is not necessarily implemented throughout the whole enterprise, implementing DevOps in a department creates a separate entity with separate governance and risk management mechanisms which are comparable to those of an enterprise.

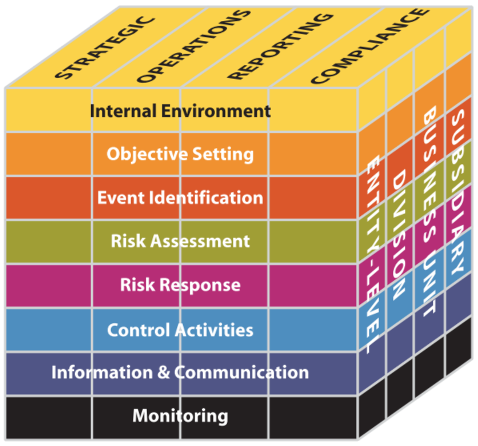


Figure 4.2: COSO Enterprise risk management Framework [4]

4.2 INTERNAL ENVIRONMENT

According to COSO [4], the internal environment is the basis for all other components of the ERM framework. It encompasses the culture within an organization and influences the risk consciousness and -appetite of its people. Relevant factors include risk appetite, ethical values and assignment of authority and responsibilities.

An important aspect of the internal environment within DevOps which is often mentioned in literature is the culture. Traditionally, development and operations have different cultures which need to be replaced by a common mindset and values. The adoption of the DevOps culture is essential for a successful implementation of DevOps and will lead to failure of the transformation if not achieved [2]. A good DevOps culture is based on respect, trust and open communication and reinforces collaboration between team members [33]. These cultural changes towards DevOps can best be achieved by promoting learning and experimentation [2]. Farroha and Farroha [13] also stress that DevOps teams should treat failure as a learning experience "not to be learned more than once". Teams should therefore focus on recovering fast from mistakes instead of not making any.

Other important aspects of the internal environment within DevOps are mentioned by Wiedemann [53] who researched general governance mechanisms in the form of structures, processes and relational mechanisms that lead to successful implementation of DevOps. She concluded that DevOps teams should be given the freedom to be able to take over all the tasks of a given software delivery cycle and should have great autonomy with regards to decision making. Since DevOps is a very decentralized concept, teams also need highly implemented communication and knowledge sharing opportunities. Within an organization using DevOps, the IT team moves from being a service provider towards being a partner of the business. Another important mechanism is therefore the assignment of a product owner who interacts regularly with the business side and is responsible for the generation and validation of requirements. In order to ensure a successful transition towards implementing DevOps, the employment of an agile coach has proven successful for organizations.

The importance of governance mechanisms in the context of DevOps is also supported by Muñoz and Díaz [32] who implemented DevOps in a Mexican datacenter in order to achieve a development that reduces the time of release. They considered governance to be a supporting mechanism of quality assurance which was based on the OWASP Software Assurance Maturity Model. Furthermore, they divided the employees into teams and assigned them specific strategic roles and responsibilities. The teams that were needed in this process are the development team, a revision control system team, a quality assurance team and a release management team. The teams were responsible for different stages of the deployment process. Wiedemann [53] supports the importance of clear roles by stating that assuming agile roles and responsibilities are essential in effectively governing DevOps teams.

Due to the focus of DevOps culture on experimenting and recovering fast from failure instead of not failing at all, DevOps culture somewhat encourages risk taking and increases risk appetite in the internal environment. However, DevOps culture can also potentially decrease operational risks due to the close collaboration of development and operations. Within DevOps, the team usually shares responsibility for development and operations of a system. Developers are therefore more likely to build a system with operational risks in mind and preventing them as much as possible.

4.3 OBJECTIVE SETTING

Setting objectives is a prerequisite to risk identification, assessment and response. Without objectives, no risks that threaten the achievement of these objectives can be identified. The COSO framework defines four objective categories which are *strategic objectives*, *operations objectives*, *reporting objectives* and *compliance objectives*. Besides describing these objectives, we also name some strategies to achieving them in this section.

4.3.1 *Strategic objectives*

Companies should start by setting their business goals at a strategic level based on the organization's mission and vision. Furthermore, the objectives should also reflect the risk appetite determined by the organization [4]. The identified literature does not explicitly mention strategic objectives in combination with DevOps since these vary heavily depending on the company setting the objectives. The only generic strategic objective is mentioned by Farroha and Farroha [13] who state that the overall strategic DevOps objective is *"to maximize investment outcome and ensure that customers continuously get increased service quality and features in a manner that satisfies their needs"*.

4.3.2 *Operations objectives*

Operations objectives relate to the effective and efficient use of the entity's resources [4]. Due to the increased speed, quality and agility which DevOps brings about if implemented correctly [9, 53], implementing DevOps processes can contribute significantly to achieving these objectives. The identified literature does not mention operations objectives explicitly; however, multiple authors suggest the use of CMMi maturity model and ITIL best practices in combination with DevOps which are helpful frameworks to implement and improve IT processes and services and subsequently ensure achievement of operations objectives [33, 32, 35]. Phifer [35] shows how ITIL processes and the CMMI engineering lifecycle fit into the DevOps process as shown in Figure 4.3. However, he also notes that implementing the CMMI and ITIL processes does not necessarily mean that a company will not encounter operational problems, the result is mainly an alignment of processes and IT needs and therefore helps realizing enterprise objectives.

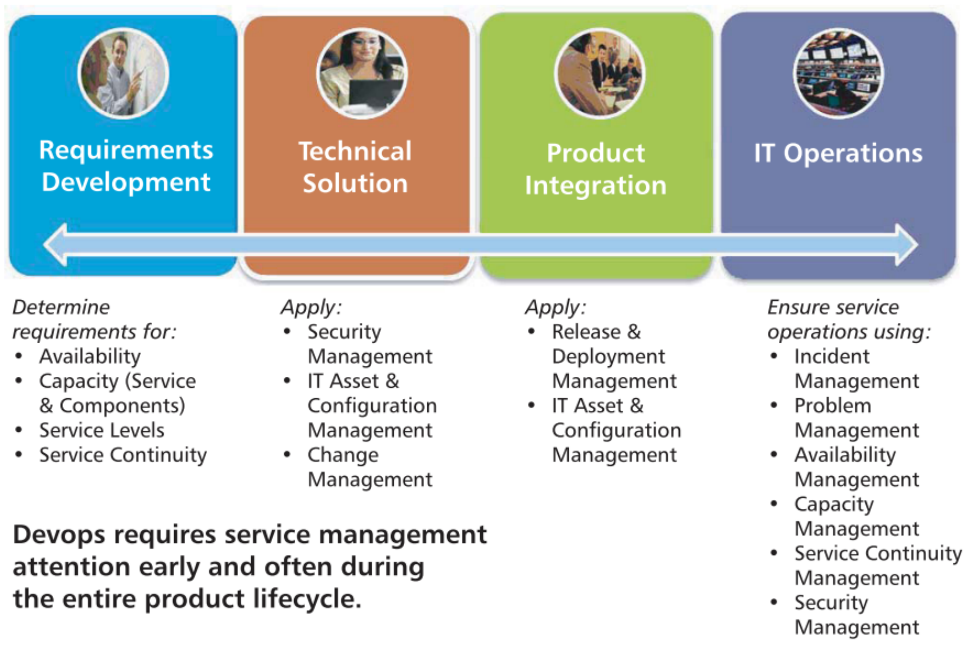

Requirements Development
Technical Solution
Product Integration
IT Operations
Determine requirements for:
• Availability
• Capacity (Service & Components)
• Service Levels
• Service Continuity
Apply:
• Security Management
• IT Asset & Configuration Management
• Change Management
Apply:
• Release & Deployment Management
• IT Asset & Configuration Management
Ensure service operations using:
• Incident Management
• Problem Management
• Availability Management
• Capacity Management
• Service Continuity Management
• Security Management
Devops requires service management attention early and often during the entire product lifecycle.

Figure 4.3: CMMI lifecycle and ITIL processes for DevOps according to Phifer [35]

4.3.3 *Reporting objectives*

Reporting objectives refer to the creation of reports that facilitate management's decision making and monitoring but also external reports such as financial statements. Multiple papers mention the integration of logging applications into the delivery pipeline in order to ensure adequate reporting of events [43, 28]. This way, DevOps can facilitate the creation of operational reports that inform management about the quality of their processes. Reporting activities that do not concern operations directly are not mentioned in the literature and do not seem to be affected by the implementation of DevOps.

4.3.4 *Compliance objectives*

The objectives which are most heavily impacted by DevOps and which are mentioned by far most often in the identified literature are compliance objectives. Companies are often required to achieve compliance with standards and laws that intend to reduce risks and create a traceable development process. Many of these compliance frameworks are designed for the traditional waterfall development process and do not fit naturally into a DevOps environment. Compliance is therefore often seen as an obstacle to employing DevOps because of required tests and controls that do not seem to fit into an automated process. Highly regulated environments usually demand segregation of duty, separated work groups and strict confidentiality as well as security measures. This contrasts DevOps where communication, collaboration and automation are central [56]. One of the main problems that characterizes this misfit is the merging of development and operations in DevOps. Developers are assigned operational responsibilities such as debugging running production systems but traditional compliance controls restrict access to production environments for developers [28]. Multiple scholars therefore advocate for a hybrid environment in which the DevOps process is integrated into the specific environment as much as possible but stays restricted by applicable regulations [56, 28].

While the examples mentioned above show compliance as an obstacle to deploying an efficient and automated DevOps process, Laukkarinen et al. [22] use the example of medical device and health software IEC/ISO standards to show that DevOps can in certain cases also be used as a helpful tool to ensure compliance. They found that DevOps was beneficial for implementing most requirements. For example, clause 5.8.6 of IEC 62304 for medical device software requires that the procedure and environment of the software creation has to be documented. In DevOps, this can easily be done with development tools such as the project management tool JIRA, source code repositories like GIT and automation software like Jenkins. Furthermore, using invariable Docker containers allow for a repeatable installation and release process which is required by clause 5.8.8. However, the authors also identified three obstacles that slow down the CI and CD procedures. Firstly, software units have to be verified which means that Continuous Integration can only happen after all units have passed unit testing. Secondly, all tasks and activities such as unfinished documentation have to be completed before the release of a software unit. Lastly, Continuous Deployment through remote updating to customer is not possible with IEC 82304-1 because the responsibility has to be transferred explicitly to the customer when taking the software into use. Whether DevOps is a benefit to achieving compliance or compliance is an obstacle to realizing DevOps therefore heavily depends on the regulations itself. In order to ensure compliance, the DevOps process in some cases needs to be slowed down or put on halt until other tasks are completed.

Laukkarinen et al. [23] concluded in a follow up paper on DevOps in regulated software environments that tighter integration between development tools, requirements management, version control and the deployment pipeline would aid the creation of regulatory compliance development practices. However the authors also note that regulations and accompanied standards could be improved to better relate regulations with DevOps practices.

4.4 Event identification, risk assessment and risk response

Event identification and risk assessment are processes which management conducts in order to identify and evaluate events that have an impact on the enterprise. While some events represent opportunities, other events have a negative impact on the achievement of priory defined enterprise objectives and therefore represent risks. Risk assessment aims at identifying to which extent these events can harm the enterprise objectives. After identifying relevant risks, companies have to decide whether to avoid, reduce share or accept these risks [4].

While some scholars claim that traditional risk management frameworks can still be used in combination with DevOps, others argue that the DevOps environment needs a new approach to risk management. Diaz and Muñoz [11] propose to add a separate risk management phase to the DevOps process in which the

ISO/IEC 2005 norm and the OCTAVE Allegro methodology are used to identify, assess and respond to risks.

Bierwolf et al. [2] argue that due to the dynamic and uncertain environment in which DevOps is mostly deployed, companies should employ a risk dialogue approach instead of a risk log, meaning that a constant conversation between employees and management about observations and information is necessary in order to identify risks. A generic risk which is often mentioned in literature to which DevOps does not yet pay sufficient attention is that of security. Some practitioners therefore advocate for the integration of security principles into DevOps which is known as DevSecOps or SecDevOps. These preventional measures are discussed in the next section.

While multiple authors suggest approaches and frameworks on how to asses risks in DevOps and argue that control activities should be implemented based on these risk analyses, risks itself are hardly named. The few risks that are mentioned are usually only listed as a justification for implementing controls, however, no complete risk analysis is conducted. There is therefore no assurance that the proposed risk responses are sufficient or on the other hand unnecessary because their corresponding risks are already covered by another control. Only DeLuccia IV et al. [6] show an example audit procedure in which a company bases its controls on three main risks to information systems which are availability, integrity and confidentiality.

4.5 control activities

Control activities are the activities which ensure that the risk responses are carried out. Although research has not defined many specific risks and responses until now, various general controls were mentioned in literature with the aim of controlling and securing the DevOps process.

Bierwolf et al. [2] use a framework to define and compare management and control measures which divides controls into four categories being *culture*, *content*, *relations* and *process*. They note that in DevOps, control measures concerning culture and collaboration are much more important than in the classical waterfall approach in which management and controls are usually focused on content and process. Because these so-called "soft controls" are already covered in the other categories (e.g. internal environment and information & communication), this section will mainly focus on the "hard controls" as means to implementing risk responses. The controls encountered in literature were grouped into six broad categories which are discussed in the following.

An overview of all controls discussed in the following section can be found in Appendix B.2

4.5.1 *Change control*

The DevOps Enterprise Forum [9] identifies change control as one major concern when it comes to DevOps practices and compliance with SOx and PCI regulations.

Change control practices intend to reduce the risk of implementing changes that lead to more failures, poor processing and unreliable systems [9]. Implementing change control is often considered to be an obstacle to running an efficient

DevOps process by many companies since manual approvals block the rapid rate of change and delivery processes. However, the DevOps Enterprise Forum claims that many of the change approvals and verifications that are usually done manually (e.g. performance testing, security scan, verification of change sets) can also be automated by defining thresholds and automated controls throughout the delivery pipeline. Furthermore, delivering smaller changes more frequently as it is the case with DevOps, reduces risks compared to large releases with many changes as it is done in traditional waterfall development. However, this claim is disputed by other scholars who claim that the high rate of delivery poses a security problem that needs to be handled accordingly [30]. In order to quickly roll back deployments and trace changes, companies should always integrate version control into their DevOps processes. This can be done by using version control systems such as Git or Subversion [33].

4.5.2 *Identity and access management and separation of duties*

Secure authentication and access management are essential to controlling the critical systems [43, 32, 28]. Another concern that auditors often mention in combination with change control and access management is the separation of duties principle which is seemingly violated in DevOps since changes can be made by a single person. The DevOps Enterprise Forum [9] however notes that it is generally inefficient to employ a strict separation of duties where two separate human beings have to make and approve changes. In some cases it is sufficient to give DevOps engineers two accounts for the different environments e.g. with administrator rights in the development environment and restricted user level rights in the production environment. It is suggested to automate the production deployment process so no person can execute the deployment without passing the automated controls first. The same procedure should be used when deploying to non-production environments. Similarly, DeLuccia IV et al. [6] show, based on a fictitious audit procedure, that the underlying concerns that lead to the implementation of separation of duty can often be solved otherwise by defining the business objectives, identifying corresponding risks and mitigating these. In order to ensure that no single person has end-to-end control of a process without a separate check point, code that is checked-in should always be peer-reviewed [6, 9, 28]. This can be enforced by signing it with personal cryptographic signatures of the developers. When the code moves through the deployment pipeline it should be automatically checked after every step of the process that both signatures are still valid and that the code has not been tampered with [9].

Multi-person authorization should be implemented in case an approved developer needs access to a specified system when he needs to fulfill operational responsibilities like troubleshooting problems. He should be able to request access via a web form and his access should then be authorized and granted temporarily by a third party, for example via a timed password or a temporary access certificate. Subsequently, an event report must be generated in which the details of this event are recorded [9, 28]. Although a strict separation of duties is therefore mostly not necessary, some traditional controls still demand this.

In this case another multi-person authorization approach has to be used when making and approving changes.

4.5.3 *Compliance*

As mentioned earlier, compliance is often seen as hindering the DevOps process. However, multiple controls have been suggested in literature to achieve compliance while automating as many functions as possible. Firstly, Farroha and Farroha [13] stress the importance of enforcing regular audits to discover irregularities early. Furthermore, the testing and development systems should be connected to a network that is separate from the production network [32, 28]. Applications that automatically test for and report compliance violations should be integrated into the process. They should terminate access if a threshold is exceeded and initiate alarms if a policy is not accepted [13].

Many norms demand that software items can be traced back to the requirements based on which they were developed. Laukkarinen et al. [23] therefore propose to introduce item tracking from requirement to the final product as a standard practice in DevOps. Software items related to requirements should be traced over the complete version history and at every point of its lifecycle. In order to enable this, workflow tools, version histories and CI tools have to form automatic connections. In order to achieve further compliance, tools should include standard templates that comply with regulations. These tools should work hierarchically by linking requirements, subsequent items and their test items and reports to each other. Lastly, the tools should guide the developer to follow the regulated workflow.

4.5.4 *Security*

Multiple papers have mentioned the integration of security aspects into DevOps in order to reduce (cyber) risks. Security is a common concern which limits the adoption of DevOps [30] and security experts have therefore investigated how to implement security practices into the DevOps process which is known as SecDevOps or DevSecOps. Mohan and Othmane [30] have performed a literature review on these terms and found that important aspects which are often mentioned in DevSecOps literature are *definition*, *security best practices*, *compliance*, *process automation*, *tools*, *software configuration*, *team collaboration*, *availability of activity data*, and *information secrecy*.

In order to ensure quality and information security, Muñoz and Díaz [32] implemented phases from the OWASP Software Assurance Maturity Model (SAMM) to structure their DevOps process and implement the right controls. The OWASP SAMM covers the phases *governance*, *construction*, *verification* and *operations* and therefore spans the complete DevOps life cycle. The governance phase is concerned with how the overall software development activities are managed. It includes security aspects like strategy, metrics, education, guidance, policy and compliance. The construction phase focuses on identifying threats and defining and building a secure architecture. The validation phase deals with testing the produced artifacts and the operations phase involves activities related to securely deploying and operating the software.

Furthermore, companies should keep an inventory of authorized and unauthorized devices and software in order to ensure platform security and version- and software management [40] and have a controlled environment for production and pre-production that separate applications from databases to protect data in case an application is hacked [32]. During the development process, incremental changes can be made directly to components if the security impact is minimal. If this is not the case, security specialists and architects have to be involved. It is therefore important to integrate automatic checks into the deployment pipeline that halt the process if necessary. Changes to untrusted data processing for example require that updated data validation code has to be developed and integrated, before the changes can be deployed [28]. Once the code is checked in, a static code analysis should be executed [6, 9]. This can for example be triggered using Jenkins. During the CI build, code should be screened for security vulnerabilities [30]. However, engineers should already pay attention to security issues during code design and peer-reviews by giving special attention to handling of unencrypted sensitive data and ensure secured access to application interfaces [9]. As a general rule, all threats meeting the security criteria must be fixed or mitigated before deployment can take place [28]. The automated deployment process should then be followed by automatic test execution [9].

The control that is mentioned most often are automated security tests [6, 9, 28, 30, 40, 43] which have to be integrated into the deployment pipeline. The testing phase of DevOps should not only focus on performance testing but should include security tests like penetration testing, and network testing and scanning [28]. Mohan and Othmane [30] refer to a presentation which advocates that companies should consider the depth of their dynamic and static security scans in the CI build chain as well the intensity of the scans and the consolidation which is the effectiveness of handling the findings. Another important control is configuration management. Hardware and software on servers should have secure configurations by using configuration management services that help rolling configurations of operating systems and application components out to all systems and keeping them in sync [40] (see Section 2.1.2). Shackleford [43] adds to this that companies can implement applications that detect whether configurations have been changed and automatically roll them back to the desired state. He demonstrates how these controls can be integrated into the DevOps process for example by using automated configuration management tools and IaC. Robinson [40] also mentions container technologies like Docker as a secure way of ensuring correct configurations in testing environments. Companies however have to pay special attention to whether the containers they use contain outdated software or other vulnerabilities when using images from publicly available repositories.

4.5.5 *Monitoring and logging*

Multiple papers mention the integration of process monitoring tools into the deployment pipeline in order to minimize risk and create reliable reporting which can be used by auditors. In case of a problem or if compliance conditions are not fulfilled, these tools can halt the deployment process and alert the developers. Nielsen et al. [33] even define monitoring as an integral part of the DevOps concept. Continuous monitoring is not just a measure to reduce risk but also allows the team to continuously improve their processes. Farroha and Farroha [13] suggest that metrics such as *mean time to repair (MTTR)* and *mean time to restore service (MTRS)* are more important to track than the *mean time between failures (MTBF)* in most cases because DevOps teams should focus on learning and moving on from mistakes instead of not making any. Other metrics they suggest are *overall process quality*, *cost of development*, *cost of maintenance*, *accessibility*, *reliability*, *interoperability*, and *availability for audits*. Moreover, security measures such as unauthorized exposures and the capability to automatically prevent unauthorized access from internal and external sources should be taken into account. However, companies should also consider user friendliness metrics like *ease of use*, *automation of processes*, *speed*, and *continuity*. Lastly, tools should not only monitor and automatically report incidents such as compliance breaches [13] but should also continuously perform logging to create traceable processes and valid audit trails.

4.5.6 *Others*

Controls that could not be assigned to one of the previous aspects were classified as "Others". In order to allow for rollbacks and ease the release process of software, a "BlueGreenDeployment" pattern can be applied: Companies should have two production environments (a "blue" and a "green" one) of which one is live. During a release, the software is deployed to the other production environment. Once it is running there, this environment including the release should go live and all incoming requests are routed to this production environment. In case of problems, a rapid rollback can be performed by just putting the first production environment live again [9]. Furthermore, Michener and Clager [28] advise that companies should organize their development activities around a threat model which they maintain. This threat model has to be updated if the threat landscape changes which can happen for example if changes with impacts on security or data processing are implemented. Lastly, companies should have backup policies and contingency plans for restoring lost data [32].

4.6 information & communication

Within enterprise risk management it is essential that relevant information about events and activities is delivered to personnel in order to allow them to carry out their risk management and other responsibilities. DevOps has an inherent emphasis on communication. Nielsen et al. [33] acknowledge this importance and developed a DevOps knowledge sharing framework which increases awareness about the different ways in which knowledge can be shared and which helps companies assess their fulfillment of important DevOps elements. They see the main challenge of DevOps as closing the gap between development and operations which requires a detailed implementation framework. A crucial element of this framework is the collaboration and knowledge within and between DevOps teams. Sharing knowledge happens through *socialization*, *externalization*, *combination* and *internalization*. It is important to note that this framework does not only focus on sharing explicit information about activities and events for better risk management but also focuses on knowledge sharing which leads to more skilled employees in general. Furthermore, Muñoz and Díaz [32] mention Scrum as a suitable methodology for DevOps that allows regular interaction between team members. Wiedemann [53] supports this statement by noting how a product owner can also bridge the communications gap between the Scrum team and the IT department.

4.7 monitoring

The monitoring category encompasses ongoing monitoring activities of the other components and separate evaluations. COSO [4] suggests to do this using internal and external auditors. While DevOps has a strong focus on monitoring operations and continuous improvement, no paper suggested monitoring the effectiveness of risk management processes such as risk responses and controls itself.

4.8 discussion

As identified by Bierwolf et al. [2], DevOps has a strong focus on soft controls. It is therefore not surprising that DevOps inherently contributes to the "softer" categories of ERM such as the internal environment and information & communication. However, DevOps can also substantially contribute to achieving operational objectives and minimizing operational risks through automated testing and continuous monitoring. DevOps can either be a benefit or an obstacle to achieving enterprise objectives, especially compliance objectives. In certain cases the DevOps process needs adjustment or has to be molded into a hybrid environment in order to remain compliant which potentially slows the process down and causes DevOps to lose some of its benefits [28, 56]. During the literature search it became evident that much of the available literature views DevOps as a purely technical approach focused on automating the development process as much as possible but substantially less research has been conducted taking the central philosophy and cultural dimension of DevOps into account.

DevOps is often said to minimize software development risks through better communication and more frequent delivery. However, especially the last claim is disputed. Although companies using DevOps seem to be generally positive about the approach [12], no empirical research including quantitative data was encountered during this review showing that companies which use DevOps have reduced their amount of failures. Furthermore, risks are not always considered and dealt with accordingly. Little attention has been given to explicit risk management in DevOps until now because traditional risk identification, assessment and response actions are not usually a part of DevOps. However, one case study has shown that risk management can still be conducted in DevOps using existing frameworks like the ISO/IEC 2005 norm and the OCTAVE Allegro methodology. Nevertheless, management should be aware of the dynamic and uncertain environment it operates in and should engage in a continuous dialog with employees to identify changes in the threat landscape [2].

Although literature has shown that it is possible to conduct risk analysis in DevOps, only few risks have been named explicitly. The controls that were mentioned by authors are often not based on earlier identified risks but are rather suggested with the intention of bringing some kind of structure and control to the process or trying to achieve compliance with norms that do not always fit a DevOps process. Risks are only named occasionally to justify certain controls which gives no assurance that the whole process is secured. Furthermore, no attention has been given to monitoring these controls and risk responses so far.

The second issue that was identified in this review is the achievement of compliance objectives. Researchers have suggested various measurements to achieve compliance with software development norms and operational frameworks, some of which hinder the DevOps process. The real problem however, seems to be that the norms and laws are not designed for DevOps and agile ways of working. While the objectives of these frameworks are clear and important, the controls they demand are designed for traditional software development methods. It has been shown how DevOps can achieve many of these objectives using a different set of controls for example by letting developers peer review their code and implementing automated, secured controls instead of demanding a strict separation of duty. Institutes such as ISO and IEC but also best practices like ITIL should therefore consider adjusting their requirements to suit more contemporary ways of software development.

4.9 CONCLUSION

The multivocal literature review has shown that there is a lack of empirical and validated research on risk management in DevOps which emphasizes the relevance of this study. While most papers have not conducted thorough risk analyses for DevOps, literature has provided a substantial amount of automated controls that can be used by companies to control their processes. However, no elaborate strategies for addressing risks and choosing controls were presented. The literature review has therefore contributed foremost to assessing the current state of research in this domain and to answering research sub-question 2. The empirical component of this research will have to give a more detailed impression of types of risks that companies using DevOps are dealing and whether these or other controls can be used to mitigate these risks.

The content of this chapter was adapted from a previously issued report by the same author [37]

5 RESEARCH METHOD

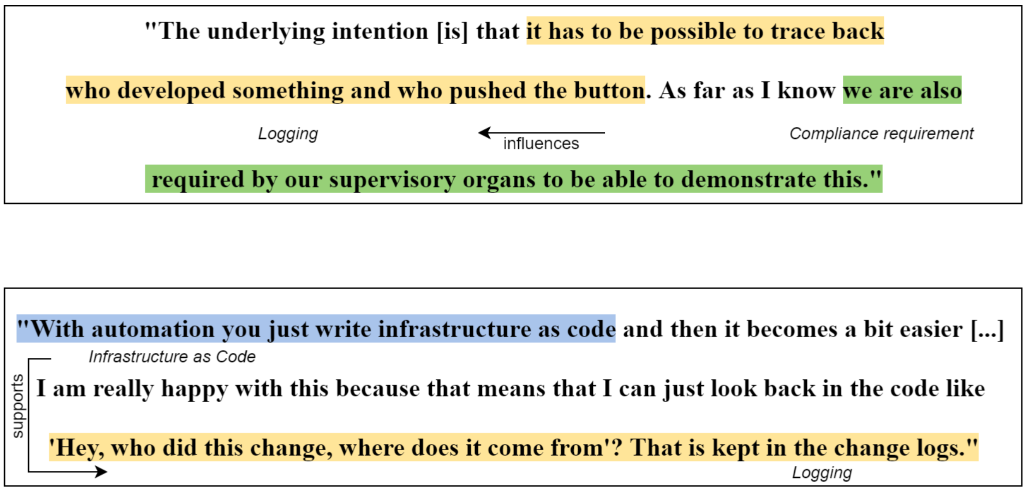
5.1 CASE STUDIES

In order to address the lack of empirical research in the field of risk management in DevOps, it is imperative to conduct an empirical investigation in the context of this research. This will increase the reliability of conclusions and allow us to base the risk management framework in a real-world context. It was therefore decided to analyze processes related to DevOps in multiple companies and to incorporate the results into the framework design. The case studies took a multiple-case, holistic design, following a replication logic [57]. This means that the DevOps risk management process was the global unit of analysis per company and that multiple companies were analyzed in order to see whether the observations and conclusions were comparable. Evidence was collected through multiple sources:

- *Semi-structured interviews*: Interviews served as the main input for the case studies. In every case study at least one interview was conducted. This amounted to 12 interviews with in total 17 employees. Interviewees were selected based on their experience with DevOps processes in their company and their position. Both low-level views from developers as well as high-level views from managers or auditors were included. Interviews were held either individually or in sessions with multiple people. One interview was conducted via telephone. After the first part of the interview, respondents were shown important results from the literature review and were given the opportunity to give their opinion about these. This was done at the end in order to not influence their answers in the first part of the interview.
- *Additional documentation*: In two cases additional documentation was used for a deeper understanding of the organization and organizational processes.
- *Observations*: Five case studies included a guided tour through the company. Striking observations were written down after the visit and included in the analysis.
- *Informal Conversations*: Remarks made from employees or people connected to the company during lunch breaks, before or after the interviews as well as during exploratory meetings were taken in to account as long as these remarks did not concern confidential information or personal opinions that were made off the record.

5.1.1 *Analysis of evidence*

All interviews were recorded with permission of the respondents. After the sessions the recordings were transcribed manually aiming to be as complete as


"The underlying intention [is] that it has to be possible to trace back who developed something and who pushed the button. As far as I know we are also required by our supervisory organs to be able to demonstrate this."

"With automation you just write infrastructure as code and then it becomes a bit easier [...] I am really happy with this because that means that I can just look back in the code like 'Hey, who did this change, where does it come from'? That is kept in the change logs."

Figure 5.1: Coding of interview quotes and mapping of relationships

possible. In some cases, in-between conversations that did not directly contribute to answering the research questions were omitted; however, the rest of the transcript was verbatim. This amounted to an average transcript length of 8 pages per interview. The transcripts were then sent to the interviewees who were given the opportunity to give feedback. Interviews mostly lasted between 40 and 50 minutes although some interviews were longer than this. In one company where multiple employees were interviewed, additional interviews took only about 20 minutes.

The interviews and some additional documents were analyzed with the qualitative data analysis software ATLAS ti following the concept of *open coding* [3]. Paragraphs and sentences were assigned one or multiple codes that summarized the topics discussed in this section (see Figure 5.1). We followed the concept of deductive coding described by Miles and Huberman [29] by using the controls derived from the literature review as a starting point for creating codes. However, many more codes were added during the analysis of which not all of them directly related to IT controls or risks but aimed at capturing contextual information which might have an influence on risks and controls. During and after the open coding process, the codes were rearranged, in some cases merged or deleted if they were redundant and classified into categories. These categories served to identify the high-level processes that were important in managing risks in DevOps. The total amount of codes left was 86. Atlas ti also allows the researcher to define relationships between codes, e.g. by defining whether one factor causes another factor as shown in Figure 5.3. This process of creating broader conceptual categories based on identified relationships between codes is also known as *axial coding* [3]. In special cases, a code was assigned to more than one category if the concept contributed to multiple categories. Some large categories were divided into subcategories.

An overview of all codes and (sub)categories designed during this process is given in Appendix C.

The relationships between code categories were inherited from the relationships between singular codes within their group using inductive reasoning (see Figure 5.2; the code *compliance requirements* was not assigned to a larger group). These categories, concepts and relationships were then put into a concept map [29] which is presented in Chapter 6. However, it is important to note that due to

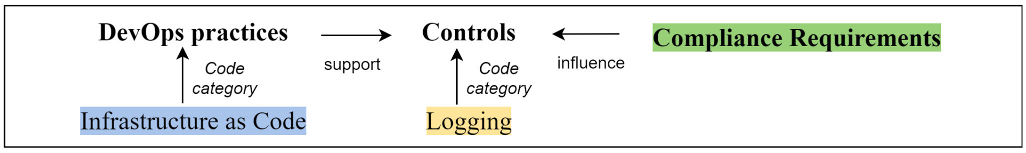


Figure 5.2: Categorization of codes and inheritance of relationships

the nature of inductive reasoning, it is possible that not all relationships between categories are completely defined. In the example given in Figure 5.2, this could for example mean that there are other DevOps practices that do not support or even hinder the implementation of controls. The relationships can therefore only be assumed to be partially true. For this reason, the nature of the relationships between code categories was not included in the concept map.

Information obtained from other sources was processed after the first coding process and added to the codes they fit best.

5.2 VALIDATION

Venable et al. [50] describe two ways to categorize validation methods based on which they suggest different strategies for validating design science research. The first distinction is made between *formative* and *summative* evaluations. The goal of formative evaluations is to improve the artifact based on the outcome while summative evaluations intend to create an understanding of the artifact in different contexts. The second distinction Venable et al. [50] mention are *artificial* and *naturalistic* evaluations. Artificial evaluations include simulations and laboratory experiments whereas naturalistic evaluations intend to explore the artifact in its real life environment.

The validation strategy applied to this design science research is based on the *"Technical Risk & Efficacy"* strategy suggested by Venable et al. [50]. This strategy is especially suitable for validating artifacts whose major design risk is technically oriented as it is the case with a risk management framework. The Technical Risk & Efficacy strategy emphasizes artificial, formative evaluations in the early stages and moves to more naturalistic, summative evaluations near the end of the design process. In our research, the artificial, formative evaluations are conducted through interviewing IT Risk and DevOps experts, who help to improve the artifact. A more naturalistic evaluations is done by presenting an improved version of the artifact to the case study respondents who can then give feedback on the expected effectiveness and efficiency of the framework in their own company. The main evaluands to be tested during evaluation where the effectiveness in terms of risk mitigation as well as the frameworks ability to allow for an efficient DevOps process.

5.2.1 *Expert opinions*

The first validation of the risk management framework was by expert opinions from experts in the fields of IT Risk management, IT Audit and DevOps. According to Wieringa [54], this validation method comprises that the designed artifact is submitted to experts *"who imagine how such an artifact will interact with problem contexts imagined by them and then predict what effects they think this would have"*. The experts were selected based on their affinity with IT Risk and/or DevOps. None of the experts were priory involved in the research so they could give their unbiased view on the results.

All interviews were recorded. After the sessions, a summary with all feedback given by the expert was written down. The sessions mainly consisted of a series of discussions evolving around the concepts presented in a slide show. Because the interviews merely served to validate existing findings instead of identifying new concepts, no coding process was conducted.

Due to the formative nature of the evaluations, the process took an iterative approach: During the interviews, experts were presented with slides summarizing the research process and the available results. After the interview, the framework was improved according to the received feedback. The improved version was then presented to the next respondent.

5.2.2 *Case study respondents*

16 case study respondents were sent a survey with a version of the risk management framework which was created towards the end of the validation process. The survey contained questions regarding the usefulness of the artifact as well as the effectiveness in terms of risk mitigation and in how far the proposed strategies would hinder an efficient DevOps process. The survey consisted of open questions and closed questions that could be ranked on a five point scale. No questions were obligatory and respondents were asked to only give feedback where they deemed necessary. The respondents could decide whether they wanted to disclose their company or remain anonymous. The survey was answered by six employees from at least four different companies.

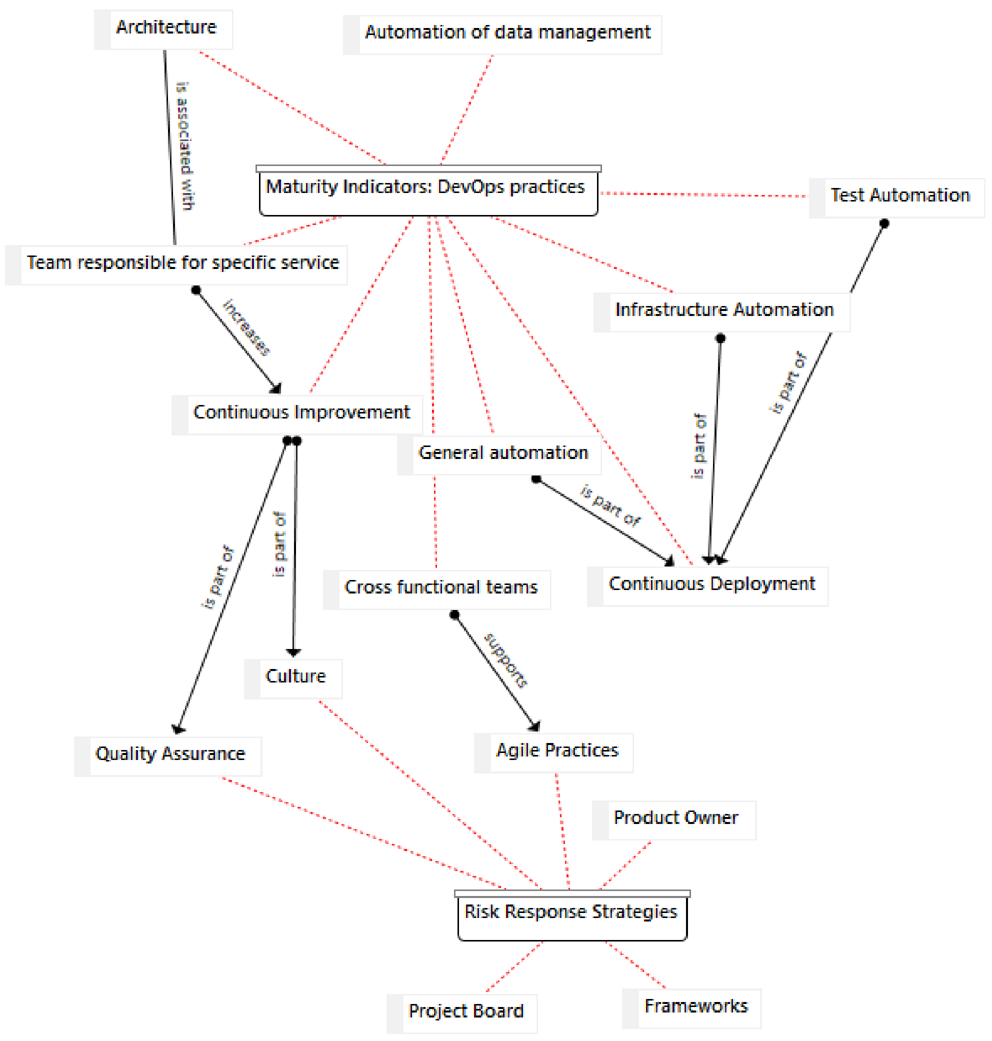


Figure 5.3: Defining relationships in ATLAS ti

6 CASE STUDY RESULTS

6.1 SUMMARY OF CASE STUDY COMPANIES

In the following section we will shortly describe each case study company. Most respondents mentioned that the DevOps teams in their company had varying levels of maturity, the analysis in this chapter will therefore focus on the processes at team level as much as possible. A summary of all companies described in the following in given in Table 6.1.

FinTech

FinTech is a large bank which is currently running pilot projects using DevOps. The interviewee is a DevOps consultant that assists teams that want to transition to DevOps in doing so. Besides the development teams, FinTech also employs infrastructure teams.

SmartIndustries

SmartIndustries is a multinational company that builds high-tech electrical systems. Among others, it provides services to the Dutch government because of which it is subject to many security requirements. The case study team at SmartIndustry provides business applications. It is currently not using any automated practices like testing or deployment but has assigned team members both development and operational responsibilities.

GeoTech

See Section 6.6 for a detailed case study of GeoTech

GeoTech is an infrastructure provider for a type of public transportation. The case study team builds applications providing maps with geo-data of relevant terrain and maintains the database with this information. The analyzed teams have been using DevOps for one and a half years. Software teams are separate from infrastructure teams at GeoTech. The infrastructure teams are again separated based on the the type of infrastructure they maintain such as Windows teams, Linux teams or Oracle teams. The interviewees were the solution architect, a product owner and a DevOps engineer.

Table 6.1: Overview of case study companies and interviewees

PSEUDONYM	DESCRIPTION	INTERVIEWEES PER SESSION
SmartIndustries	High-tech manufacturer	- Manager Competence Center - Scrum Master
FinTech	Bank	- DevOps Consultant
InsuranceInk	Insurance company	- Manager Software Development and Manager Infrastructure & Operations
GovTech	Governmental agency	- Project Manager & QA Officer
GeoTech	Infrastructure provider for public transport	- Solution Architect - Product Owner - DevOps engineer
PublicServices	Independent regulatory agency	- Internal Auditor, Software Engineer, Product Owner & Security Manager
WebSales1	E-Commerce company	- Security Engineer
WebSales2	E-Commerce company	- Risk & Compliance Officer
WebTech	Internet Agency	- DevOps Engineer

GovTech

GovTech is a governmental agency. The analyzed team maintains the website of this agency. It works according to the Scaled Agile Framework (SAFe) and is responsible for both developmental and operational tasks. The platform on which the website is hosted is maintained by a third party which deploys a release package that GovTech has prepared for production every two weeks. The interviewees were a quality assurance officer and a project leader.

PublicServices

PublicServices is an independent governmental agency that is licensed by the Dutch government to provide specific services to the public. The interviewees were a software engineer of a team providing digital services to the public (in the following referred to as Team External), a product owner of a team providing software tooling to the other teams (in the following referred to as Team Internal), an internal auditor and briefly, a security officer. The services of Team External are partly run in Amazon Web Services (AWS), other services are run on an internal platform which is maintained by infrastructure teams.

WebTech

WebTech is a small digital services agency with about 100 employees. It provides digital services such as websites, dashboards and calculators to multiple clients. The interviewee is a DevOps engineer at WebTech. WebTech uses continuous deployment.

InsuranceInk

InsuranceInk is a large Dutch insurance that is currently transitioning to DevOps. The interviewees were a manager of software development and a manager of infrastructure. InsuranceInk is currently working on automating their deployment and uses tools like Azure DevOps to check in code and run some automated tests. It also uses Octo Deploy for automating deployment.

WebSales1

WebSales1 is a Dutch e-commerce company that is known for its successful development processes and strong company culture. It has been using DevOps for five years. The interviewee is a security engineer who supports the DevOps teams in ensuring the security of their products. For this he serves as a contact person for security related questions, is involved in early design stages and conducts security audits. The systems of WebSales1 run in AWS and it has automated most of its production completely.

WebSales2

WebSales2 is a Dutch e-commerce company that operates worldwide. It has been using DevOps for many years. The interviewee at WebSales2 is a risk & compliance officer who ensures that suitable IT controls are present and are correctly implemented. The company separates between development groups and administrators of assets whereby administrators are people that run the platform underneath the application. However, this is only a separation of responsibilities since both groups have the same access rights. The company is currently transitioning from one way of working DevOps to another DevOps process since it is implementing a microservices architecture. This means that teams now work with more fragmented code on which they can enforce a full review when it is to be deployed which opens up new opportunities in terms of automated testing and authorizations. WebSales2 has also automated its deployment.

6.2 OVERVIEW OF CONCEPTS

According to the process described in Section 5.1.1, a concept map was constructed. This map entails the high level categories and their codes or names of code groups and is shown in Figure 6.1. The map does not only show the risks, risk mitigation mechanisms and control categories but also includes some categories that were identified as being indirectly related to these concepts.

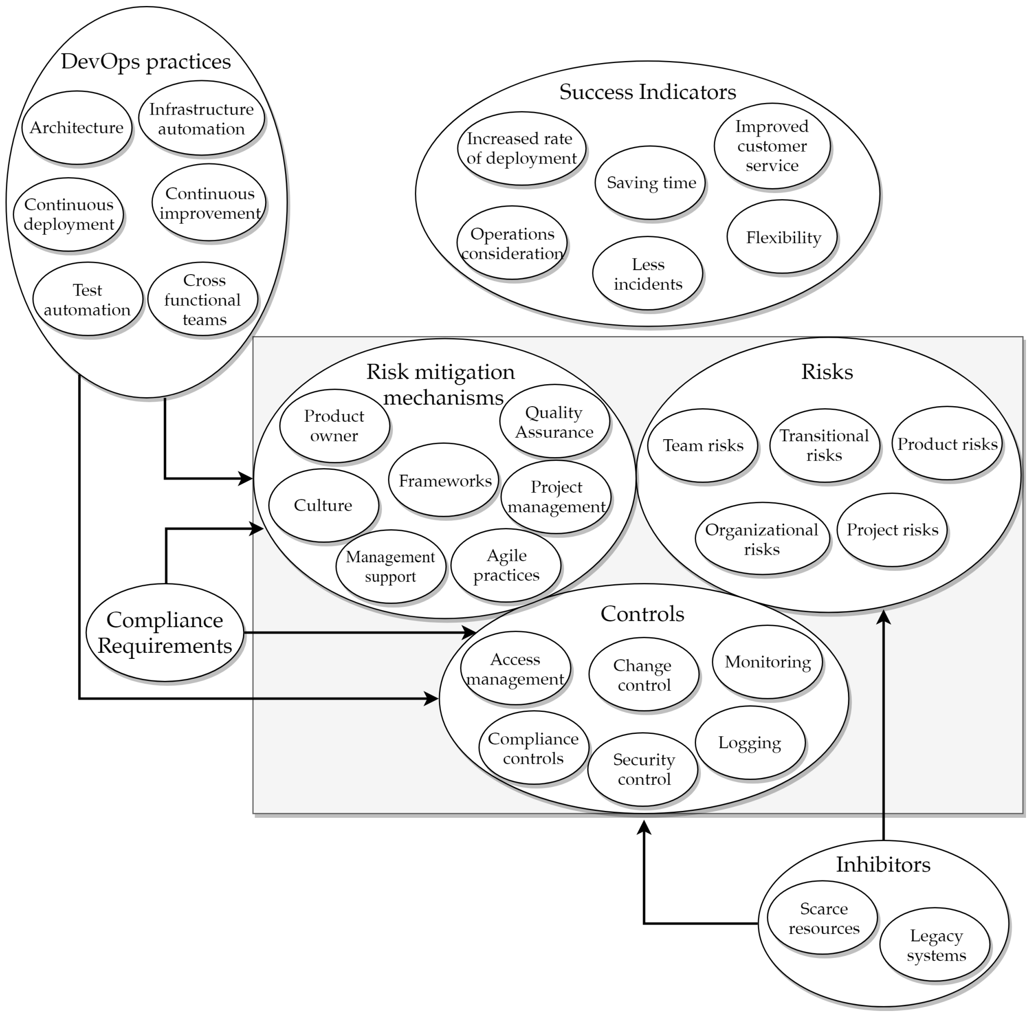


Figure 6.1: Concept map

Furthermore, the map shows the relationship between these contextual concepts and the core concepts. The contextual concepts will be elaborated shortly in the following before we describe the findings of the analysis that are directly related to the research questions.

6.2.1 *DevOps practices*

During the first part of the interviews, respondents were asked to describe their understanding of DevOps and to which extent they implemented DevOps in their company. This was done to understand the maturity of their DevOps processes and the experience the company had in applying this practice. Multiple companies indicated that they were still in the implementation phase and were currently running pilot projects. The maturity of the DevOps practices showed to have a significant influence on the nature of controls and risk mitigation mechanisms that were integrated into the process. For example, multiple companies indicated that they were not experienced enough to employ continuous delivery techniques or to rely on automated testing which made the implementation of manual change controls and deployment activities necessary. When designing a control framework, it is therefore important to consider the DevOps maturity of the respective company. There is no explicit relationship defined between DevOps practices and risks because the risks were already defined as being risks within a DevOps environment and do not seem to be further affected by the extent to which the practices are implemented.

6.2.2 *Compliance requirements*

The case study companies were subjected to various levels of compliance. In at least two cases the compliance requirements influenced the nature of controls directly. The risk & compliance officer at WebSales2 explained that PCI regulations demand a set of change requests and an approval at the deployment stage. The security officer at PublicServices mentioned that the ISO security norm they were following demands that roles are sufficiently separated and access is given according to the principle of least privilege. Other companies were subject to strict internal compliance. At SmartIndustries, every new employee needs to undergo strict security screening that takes four to twelve weeks and makes it impossible for the manager to quickly hire new people when more employees are needed.

Although some case companies seemed to be subjected to compliance requirements much more than others, not all respondents thought that these requirements were hindering their process. Some respondents even stated that DevOps allowed them to be more compliant because new work items that were derived from compliance requirements could be put on the backlog and be picked up directly in the next sprint. As shown in Figure 5.2, some DevOps practices also supported compliance requirements e.g. through automated logging and version control. Many respondents also stressed that due to the General Data Protection Law they had to be very careful with the protection of personal data, however, they did not think that this was related to the DevOps process.

6.2.3 *Inhibitors*

Two factors were categorized as inhibitors because they were mentioned by respondents as hindering an effective implementation of controls or as increasing risks. Firstly, an employee mentioned that they had too much work for the amount of employees in the team which leads to a large backlog and and certain work like patching being done less frequently. He also mentioned that this made it difficult for them to employ a strict separation of duties or peer reviews. The second inhibitor that was encountered were old systems. Several respondents indicated that their mainframe was still running with COBOL which for example made it difficult to perform rollbacks. Naming these inhibitors does not necessarily imply that companies facing these problems will not be able to conduct effective risk management. However, having scarce resources or systems that do not sufficiently support automation and agile working does likely require the implementation of additional controls that are not desirable in a DevOps environment since they will slow the process down. Addressing these inhibitors is therefore seen as a prerequisite to our framework.

6.2.4 *Success indicators*

Before identifying risks and controls in the analyzed processes, it was necessary to identify the particular outcomes that a company had achieved with their DevOps process. This was required so that one may identify those controls that would burden the process as little as possible. Respondents were therefore also asked to elaborate on the influences that the DevOps implementation had on their organization and operation. While negative influences were mostly categorized as organizational risks, positive influences were rated as indicators of a successful DevOps implementation. Not surprisingly, it was shown that many of the success factors related to codes in the DevOps practices category. For example, an increased rate of deployment was mostly due to the automation of processes. These success factors were included in the concept map and served as a reference to identifying which controls would still allow for a successful DevOps implementation. All companies reported some positive outcomes of their DevOps transition.

The success indicator *Less incidents* was added because a case company provided quantitative data to demonstrate that they had managed to lower their number of incidents significantly by implementing DevOps. However, a risk and compliance officer at another company disputed this finding by stating that a company in which she worked earlier that did not work agile had much less incidents than her current company because this company used change requests and stages. A manager at the first company made the remark that the incidents in DevOps are not comparable to incidents in waterfall since they are of a different nature. Teams will usually notice malfunctions in their systems much earlier due to extensive testing and monitoring and will likely fix them before users can create tickets and officially declare them incidents. The remaining occurrences that are escalated to incidents therefore relate to larger issues that are already known in the team. Besides an increased rate of deployment, respondents reported that they were saving time due to the automation of manual tasks and were more flexible because of the agile way of working and less dependencies on other teams. They also thought that they could implement customer wishes better in DevOps as well as involve them in the process and inform them early in case of problems or delays. The success indicator that was mentioned by far most often was that developers were now writing code while already considering how it could be maintained best because they have to fulfill the operational tasks themselves. This ultimately leads to better code and documentation.

Although we acknowledge and define some success indicators that will aid in the creation of the risk management framework, it is important to note that conducting risk management and implementing effective controls will not directly lead to a successful DevOps process. Controls mainly serve to eliminate or mitigate the negative relationship between risks and DevOps success. Indicators that are directly related to DevOps success (besides the identified DevOps practices) are not within the scope of this research.

6.3 identified risk categories

Since risks vary per company and environment, it is not possible to create a complete list of all risks in DevOps. The risks mentioned by respondents or encountered during the case studies therefore had to be grouped and put into a wider context of risk categories to allow for a complete overview of risks that need to be considered. The risk categories will be elaborated in the following

6.3.1 *Team risks*

Team risks were mentioned most often and seem to be the biggest risk category applicable to DevOps. Many respondents acknowledged the risk that the teams could prioritize the wrong items of the backlog for example by choosing development tasks over operations tasks. They also mentioned that less important items would probably stay on the backlog for a very long time and would not be picked up which can lead to faulty or insecure products. Furthermore, teams are sometimes pressured to take quick decisions because of short delivery dates and stakeholder expectations. This can lead to taking unwise decisions that increase the technical debt or deployment of products that are not yet entirely finished or secure (see Section 6.3.2). Due to the high degree of autonomy and the required access and authorization rights, DevOps teams can work very independently. This can lead to such a high degree of team autonomy that the team activities and systems become completely intransparent to the rest of the organization and teams start to close off their products from other employees because they feel that it is their own product. This also means that other teams would not be able to intervene or even notice in case the subject team does not maintain their system properly. As one respondent put it, teams could also become so independent that their start *"reinventing their own wheel"*. One group of respondents also mentioned the risk that teams will not stick to the schedule they agreed upon at the beginning of a sprint because they start working on other small tasks which they deem important or help out other teams for a few hours. Lastly, many controls that were mentioned relate to the general risk of internal fraud. Many organizations grant their employees access to production which means they could potentially deploy malicious code without having to show it to a second person first. Furthermore, many employees have access to databases which enables them to steal confidential data.

6.3.2 *Product risks*

Product risks are risks to the confidentiality, availability and integrity of the product. Since DevOps focuses on frequently deploying small, incremental changes, the deployed product might not be completely compliant or secure at all times. One respondent mentioned in this context that agile teams often call the first deployed versions "minimum viable product" in order to legitimize a potential lack of features like compliance or security. Nevertheless, security and compliance were not mentioned as key risks which are specific to DevOps, although almost all of the respondents stressed the general need for being compliant and secure. Moreover, multiple respondents addressed the risk of poor system continuity. This risk is an output of risks addressed earlier like deploying products that are still under development or because DevOps teams tend to prioritize developing tasks above operational maintenance tasks, which might compromise the system availability. One respondent also mentioned that due to the flexible and fast way of working there were risks that the product would not always comply with internal guidelines such as communication policies or the house style. For example, his company usually expects that the communications department is involved in a new product from the beginning on which is not always possible because their scheduling is less flexible than that of the DevOps teams (also see Section 6.3.3).

6.3.3 *Organizational risks*

Organizational risks refer to risks that originate from the structure of the organization and the organizational processes. Several respondents mentioned that they had difficulties working with other teams or departments in their organization which were working in a non-DevOps way. One interviewee mentioned that the other departments on their side of the organization still works with traditional project methods and that it costs his team a lot of time to justify their own way of working. He gave the following example:

> *"[...] the risks to me are primarily in the people around it [DevOps]. Of course everybody has a good intention why they want to or don't want to use it but that creates the risks. The moment someone comes along with a traditional planning following a waterfall method and they start putting pressure on the agile teams their processes become disorganized. And that for me is the biggest risk."*

This experience was shared by another respondent who mentioned that the higher levels of his organization work with traditional methods like PRINCE2 while his team works according to business priorities instead of a traditional planning which do not always complement each other. This is especially problematic when the top management wants to obtain resources from the DevOps teams for their own projects while the DevOps teams face other priorities.

6.3.4 *Project risks*

This category encompasses risks that relate to plannings and agreements made with clients. Multiple respondents mentioned predictability as a risk of DevOps projects. Due to the agile way of working, deadlines and delivery dates are less predictable and sometimes hard to realize. This agile risk is amplified by the shared responsibility of the teams for both development and operations. Operational incidents cannot be planned at the beginning of the sprint but sometimes have to be resolved immediately once they occur. This can impact the sprint planning and lead to teams lagging behind their planning. However, other respondents mentioned that their overall predictability had increased since transitioning to DevOps because they are less dependent on other teams. This risk was mainly mentioned in combination with larger projects.

Furthermore, one respondent lamented a lack of structure and clarity in large projects which are managed in an agile way. According to this respondent, he misses parts of the traditional project management methodologies like PRINCE2 which make use of impact analyses, steering committees and resonance groups. These responsibilities are less defined and visible in agile projects which makes it difficult for employees to escalate concerns and observations that could potentially threaten the success of the project. He suggested integrating some traditional project management elements into large DevOps projects.

> *"We are currently in quite a large project [...] and because it is so large I can see a lot of risks impacting the chances of success of the project and I notice that I cannot just go somewhere with this easily. Normally you have steering committees, resonance groups, you name it, but now you don't have this. In this case I do miss the steering of this."*

6.3.5 *Transitional risks*

Transitional risks only apply to organizations that have recently started implementing DevOps and still have a low DevOps maturity. Some respondents such as a security manager stressed that it was difficult for teams to realize what is meant by the idea of "end-to-end responsibility" of a product. According to him, teams were not yet fully aware of all security and compliance standards they had to fulfill and it was difficult to communicate them from an organizational perspective. Another respondent noted that the DevOps transition created a lot of stress among his employees because of their increased responsibility which not all of them were happy with. A third respondent found that operational engineers often felt redundant after teaching development teams how to do operational tasks which could lead to unmotivated employees. As evident, all transitional risks also fall into one of the other risk categories (in this case all identified risks are team risks; nevertheless, it cannot be ruled out that more transitional risks exist that fall into another category). However, the difference between transitional risks and the other categories is that transitional risks disappear once the company has successfully implemented DevOps and employees have become accustomed to it.

6.4 GENERAL RISK MITIGATION MECHANISMS

This section introduces some some general risk mitigation mechanisms for addressing risks in DevOps that have not yet been translated to specific controls.

6.4.1 *Product owner*

In order to address problems relating to the prioritization of backlog items, respondents stressed the importance of a capable product owner who carries the overall responsibility for the realization of tasks and connects the team to the rest of the organization and the clients. The product owner has to monitor the execution of operational and development tasks and can determine the urgency of an unexpected incident as well as when it should be solved. In multiple companies, the product owner was also the role deciding whether an item fulfilled the stakeholder requirements and could authorize it to be deployed into production.

6.4.2 *Agile practices*

All companies used an agile working process. One company had specifically implemented the SAFe framework and claimed it helped them maintain an overview of risks and create countermeasures. A risk management approach was integrated into the process starting with risk evaluation sessions and documenting and refining user stories. Risks were then considered in writing user stories and Definitions of Done.

6.4.3 *Quality assurance*

One case company demonstrated the use of a Quality Assurance officer who was supporting and monitoring the DevOps process from a high-level view. While the teams are mainly responsible themselves for the quality of their products, the QA officer checks general quality criteria such as whether the Definition of Done is correctly defined. He is also involved at the beginning of each project where he participates in a risk evaluation session with various other disciplines and ensures that other roles such as architects are sufficiently consulted. The QA department also ensures compliance and other quality aspects of the products. Once the high-level requirements of a new project are made sufficiently transparent, responsibility is transferred to the DevOps teams who can then start defining epics and user stories from these. They also used the code quality tool SonarQube to ensure quality.

6.4.4 *Project management*

As mentioned in Section 6.3.4, one respondent addressed the need for traditional project management elements in complex agile projects. These project might benefit from appointing steering committees, project leaders and assigning risk officers whom people can approach with concerns. An example of how this could be done was demonstrated by SmartIndustries: Large projects that exceed a certain number of hours are divided into gates for which a plan is created. For these large projects a steering group and a program manager are assigned and risks are defined and explicitly monitored.

6.4.5 *Frameworks*

Frameworks that were encountered in combination with DevOps were as follows: One company used the DASA model which was introduced in Section 2.1. Few companies used ISO 27001/2, SDL or NIST for information security. Some respondents also mentioned using ITIL and wanting to maintain the ITIL processes when switching to DevOps. One company explicitly mentioned using COBIT, however, they only used it for critical applications that are relevant for requirements from the Sarbanes–Oxley Act. As mentioned in Section 6.4.2, one company used the SAFe framework for general process management.

6.4.6 *Organizational culture*

Another general mechanism that was encountered was the organizational culture. As also found during the literature review in Chapter 4, this element is the most difficult to implement, yet one of the most important elements. The organizational culture is connected to many controls and risks that are influenced by this. The case company with the most mature processes also had the most visible culture, which is also known outside of the organization. According to the respondent, most teams felt responsible for their own product and wanted to continuously improve it. Furthermore, he stressed the importance to create a culture in which people welcome change and dare to speak up if they made a mistake:

> *"We have something here which we call 'failure-treats' [...]. You take them to people to talk about your failure. Funny enough, in IT you mostly see people saying 'Yes, I did that once too [...] That is really stupid but I can think of three places where that could happen to us too'. And then everybody can laugh and it's all great. [...] It creates a much friendlier, open atmosphere. Because the ugliest thing from a security point of view is if people don't dare to speak up. If there's a taboo about that, especially if it's about the reputation of people. That can stand in the way of openness."*

6.4.7 *Management support*

The last factor that was identified in the context of risk mitigation mechanisms was the importance of management support for DevOps. Teams require a great deal of autonomy and decision making capabilities (see Section 6.5.7) in order to work effectively and act quickly upon incidents which requires management to give this control to the teams. Management support was shown to not just be a very important risk mitigation mechanism but should rather be viewed as a prerequisite for using DevOps effectively. A lack of management support can even lead to team and organizational risks.

6.5 identified controls

The identified controls were again grouped into categories. For this we used the categories encountered during the literature review in Section 4.4. A category "soft controls" was added to this list.

6.5.1 *Change control*

Team responsibility and communication

In general, companies that deployed frequently gave teams more autonomy when deploying these changes. Especially companies where changes had a relatively low overall impact like WebSales1 and 2 as well as WebTech gave teams a lot of freedom in assessing the impact of a change themselves and taking appropriate measurements. A frequent expectation mentioned was that teams should communicate themselves with other teams if they were about to implement a change that might impact other systems. Some companies let their team deploy small changes themselves if they fulfill a set of criteria. For example, InsuranceInk requires them among others to have a plan for rollback, assess the impact and talk to the product owner.

Change approval

For a more elaborate description per company on the authorization process and SOD principles see Section 6.5.2

Most teams have integrated some form of change approval into their process. The rigidness of this approval process however varies per company: At SmartIndustries, most changes have to be registered for authorization of a Change Advisory Board (see next section). Only very small changes such as patches can be installed directly by the team themselves. In many companies the product owner is the last person authorizing a change because he needs to validate whether the requirements he designed are implemented correctly. GovTech has outsourced its platform to a third party that does not want to grant the developers access to production which is why all deployment activities are still conducted by the hosting organization. The official approval is also given by the product owner before changes are prepared for a release. FinTech has recently designed a new change approval process in the light of their DevOps transition: While changes previously had to be approved by a separate department, changes that solely affect the DevOps team itself, now only have to be approved by the product owner of the team. Changes that also affect other applications and teams have to be approved by another person from the affected team. The respondent stressed the importance of chain alignment in this context as teams have to be aware which other systems are impacted by changes to their own system.

WebSales2 previously deployed changes directly to production after a minimal amount of testing. The official approval was then given after deployment by another team member (see section *Post-deployment testing*). The new process to which WebSales2 is currently transitioning allows more automatic testing upfront and also enforces an authorization by another developer before deployment. At WebTech, changes are also mostly approved by a more senior developer.

Change advisory boards

Multiple companies use a Change Advisory Board (CAB) in which representatives of different business units are involved. The CABs however take varying roles in the companies: At PublicServices and WebSales1, the person committing the change himself decides whether a CAB is necessary. The CAB then takes an advisory role by determining risks and impacts of the change and decides which actions need to be undertaken before deploying a change. According to the respondent at WebSales1, the involved people will especially look at risks such as privacy and security. InsuranceInk employs a change manager who guards the change process and facilitates it with a technical change advisory board (tCAB) which is involved with technical changes. Besides the tCAB, InsuranceInk also has a regular CAB. At SmartIndustries, the CAB meets every two weeks and takes an authorization role. When submitting a change, developers then have to fill in an Excel Sheet in which they answer some questions and include a fallback plan. The CAB then estimates the risks of the change and makes a change calendar which it publishes.

Categorizing changes

In order to find a balance between giving teams autonomy and still keeping control over changes, companies like GeoTech, FinTech and InsuranceInk categorize their changes and treat them differently depending on their category. At GeoTech, developers have to assign a change category to the change themselves when registering it whereby 0 is the lowest category. Changes of category 1 and higher are checked by a CAB that authorizes all changes on an organizational level. The teams can then deploy the changes themselves.

Different to GeoTech, teams at FinTech cannot determine the category of the change themselves. The interviewee expressed the intention to automate changes of a lower category in the future. Higher category changes will always have to be approved by people inside or outside of the team depending on the impact. InsuranceInk distinguishes between categories like *quick releases* (small changes), *standard releases* and *speed releases* (changes that have to be deployed immediately). Standard releases have to be assessed for impact and in some cases have to be authorized by a CAB. Changes of a technical nature are always checked by a technical CAB because they have a bigger impact. Many of the regular software changes are currently left to the team responsibility. SmartIndustries does not officially use change categories. However, the manager noted that in their JIRA Scum board, requirements are broken down into items like *sub-task*, *task*, *story* or *epic* which naturally gives the developers an indication of the size of the change.

Version control

All companies stated to use version control. SmartIndustries has recently deployed Git as a version control system while WebTech uses Gitlab. The software engineer at WebSales1 stated that version control made it very easy for him to trace back who conducted a particular change. He was particularly positive about versioning configuration sets that were written down as code. Furthermore, one respondent mentioned that they also versioned their Docker containers in which the applications were running which demonstrates the use of IaC and containers for versioning both infrastructure and applications.

Change registration

Besides using regular version control, some companies still require changes to be registered by the developers. Multiple tools were mentioned for documenting the process. While FinTech uses ServiceNow, SmartIndustries used JIRA. Although documentation is still necessary, most companies stressed that the overall amount of documentation had significantly decreased since using DevOps and that changes can be deployed much faster now. According to a respondent at PublicServices, the change request forms in their company are nowadays more used for communicating changes with each other than for documenting the process. GovTech also uses JIRA and saves documentation in an internal wiki when deploying changes. When making a change, teams have to fill in a release form which is saved to this wiki. After this, automated tests are deployed and the test reports are also saved in the wiki. The wiki is generally used for administrative documents and comments while JIRA documents the agile process. GeoTech uses a central changesystem called ICD which is produced by IBM. This is also required for operational changes such as creating accounts.

Automated testing

As part of automating the process, many companies are currently aiming to automate the testing process. This eliminates manual tasks and increases the deployment process but also ensures that crucial tests are always run when making a change. While companies like WebSales1 and WebTech already use CD with extensive automated tests, other companies created a hybrid environment with partly automated and partly manual tests or used CI. Team External at PublicServices still employs testers but also uses automated regression tests that run every night. Furthermore, a piece of code can only be checked in when a change meets the requirements. This means changes cannot be committed if the code coverage of tests is less than 70% or the build is broken. GovTech and GeoTech have also automated substantial parts of their testing process although.

Post-deployment testing

WebSales2 had the least amount of preventive controls. Changes in their old change process were previously directly deployed after a minimum amount of automatic testing without the approval of another person. Traffic to this new production environment was then limited to a small percentage until another team member (usually a more senior developer) conducted a peer review on the change and approved it. The traffic to the new production environment was then extended to 100%. This way of A/B testing provides WebSales2 with metrics that help them to decide whether they want to extend the change or roll back. However, the authorizing developers also reviews the code in terms of security and compliance. WebSales2 also was the only company that indicated having some automatic change notifications in the most critical parts. These controls can assess the impact of a change based on the dependencies of the system. However, these change controls only take place the moment a change is deployed which does not give the developer enough time to react before the change goes live. In other parts that are not covered by automatic change controls, teams are expected to assess the impact themselves and communicate with other teams if necessary.

6.5.2 *Identity and access management and separation of duties*

Access to production

Most questions concerning access management evolved around the question who should gain access to the production environment. Whether companies gave employees this access to production or not depended on the circumstances. WebSales2 grants its team member complete access to production in order to fulfill operational tasks like troubleshooting; however, they do have different privileges in production than in development. Concerning companies that fully automate their deployment and don't grant employees access to production, the respondent noted:

"You need to trust your system you need to trust all this deployment, the pipeline and the tests you perform before. And also it is very hard to troubleshoot if you don't know the application. So let's imagine for a moment we trust our administrators or the platform owners to troubleshoot, it is really hard. Now if you think about new technologies like containers where they don't really have privilege access but they can still troubleshoot or even trigger new versions of the pictures of applications very fast. Then again the risk becomes lower."

Developers at WebSales2 do not have default privileges to access to configure the tools in the pipeline. WebTech also gives developers access to production but still uses more traditional role descriptions and only grants this access to specific roles. On the other hand of the scale was WebSales1 which uses continuous deployment. This high degree of automation allows them to restrict developers access rights to production substantially. Employees are granted as little access as possible which effectively leads to most employees not having access to production. Teams however do maintain the pipeline themselves by adding and changing tests. The respondent at FinTech mentioned that not having to give developers access to production anymore was was their ultimate goal for the DevOps transition:

"In the end of course you want everything to go into the pipeline automatically so that you don't have to give access to anybody but in the mean time we will have to come up with some intermediate solution."

Timed passwords

Between giving employees complete access to production and none at all were companies like FinTech and GeoTech who used timed access rights. FinTech stated that it wants to keep the separation of duties principle after its DevOps transition and therefore doesn't want to give developers unlimited access to production. It is currently running pilots with the software "CyberArk" which allows developers to request timed passwords. GeoTech grants developers restricted access once a change is authorized by the CAB. The access then is logged and a standard notification is sent to the security department. While this control seems like an effective compromise, it depends on the circumstances whether this is practical. The interviewee at WebTech noted that he needed access to the production about twice per week and that requesting timed passwords for this frequent access would be rather impractical and not add much security since he had the regular opportunity to tamper with the production environment if he really wanted to.

Separation of duties

All companies said that they wanted to use some kind of separation of duties principle, although not all companies enforce this. Many case companies work with peer reviews, although the manner in which they were conducted varied. At WebSales1 and Team External of PublicServices, code reviews are technically required because developers cannot check in a piece of code if his merge request is not approved by another developer. Furthermore, at Team External the product owner is the person ultimately deploying the change.

In other teams, respondents trusted that another person would see the code automatically because people are working together in DevOps projects and have access to the same code base. This is also the case with Team Internal at PublicServices where all team members are able to deploy code to the next stage. Although it does make use of a reviewing role, the review is not technically enforced. According to the product owner, it does not happen that a piece of code is deployed that only one or two people saw because of the way in which the teams collaborate. The respondent at WebTech had a similar reasoning. Furthermore, WebTech also uses code reviews by encouraging developers to create merge requests and assigning those to a more senior developer who has to check the code and merge it. This is not necessarily done for small projects.

SmartIndustries also demands peer reviews, however, these are not enforced. Developers only have to check a to-do list in JIRA which includes a code review, although this list is not connected to the system. One of the respondents noted that access rights are not regularly reviewed and that some employees still have old access rights which they don't need anymore. At GeoTech, teams have access to all accounts because of the varying tasks they do. Access to production however has to be requested on a temporary basis after a change has been authorized.

Because GovTech has outsourced its platform to a third party which manually deploys changes every two weeks, a separation of duties principle is integrated into the process. Furthermore, the product owner needs to approve each release package. As mentioned earlier, the product owner is also responsible for approval at FinTech and Team External. InsuranceInk is currently still in transition and has not yet decided how to handle the separation of duties principle. The respondents however did note that they do not necessarily want to stick to a traditional segregation of duties principle. According to one manager, the underlying goal of this principle is to prevent team members from conducting fraud. If there is another way to prevent them from doing to they would also be willing to try this. Currently, the technical application managers deploy changes which effectively means that the separation of duties is still available.

The respondent at WebSales2 noted that although the company requires a change authorization before the code goes live in the new process, code could technically be changed again once the code is live since developers have access the production in order to troubleshoot. This demonstrates that even the most effective separation of duties controls will not mitigate the risk of internal fraud if access to production is not restricted.

Separate roles and access rights

Some companies still worked with specific roles and assigned access rights per role. A manager at InsuranceInk stated that they were considering the possibility to grant employees dynamic access rights per sprint by assigning them specific responsibilities and access rights every sprint.

Team External at PublicServices also divides roles strictly. The team still uses specific role descriptions like developers, testers and operational employees who are responsible for a specific part of the deployment process. The process follows the common DTAP principle where no employee is able to deploy a piece of code to the next environment if he deployed it to the previous one. WebTech grants employees required access to servers by using public and a private SSH keys. Employees that need access to production servers are granted those on the basis of their public key. In case they leave the company or their private key is compromised, access can be easily revoked by removing the public key from the central access list. WebTech however does not make use of temporary access rights.

Others

GeoTech has created a selfservice portal for one of its platforms in which the team can grant access rights to people to view data that is visualized in their application. However, the organization has defined data owners that have to give their consent if someone wants to access their data. Most companies also said to use separate accounts where every employee has one or more accounts to access parts of the deployment chain.

6.5.3 *Security*

Security-related monitoring activities are discussed in Section 6.5.5.

On a high level, GeoTech has divided security into three main components which are *privacy*, *security* and *company risks* such as application failures. SmartIndustries distinguishes between three similar types of security which are *physical*, *information security* and *cybersecurity*.

Team responsibility

Many companies stressed the responsibility of the teams to ensure the security of their own product. InsuranceInk follows the SDL framework level 2 and holds teams responsible for executing it correctly. As mentioned in Section 6.4, the respondent at WebSales1 thought that it was very important that team members feel responsible and dare to step up if they have any security concerns because no security auditor would be able to find all security issues. The respondent at WebTech also noted that it was necessary to rely on the goodwill of employees to some extent because too many security controls would hinder the process:

> *"Of course we have NDA's, we have contracts with [WebTech], we have agreements [...]. [But] you have to be able to deal with each other in good faith. Security is always a consideration between usability and security, the more usability the less security and the other way round."*

Another respondent mentioned in this context that very strict systems with a lot of security checks will inhibit the efficiency of the workflow which leads to team members developing workarounds that again compromise security.

Some companies even gave the teams a dedicated moment where they could mention security risks. At GovTech this happened during so-called "pokersessions" where the user stories were evaluated in terms of effort and time they cost to implement. The risks that have to be dealt with are then considered when assigning a story a "pokervalue". Large stories are split up into smaller sub-tasks to make them more manageable. WebTech also stated to conduct sessions in which security concerns as well as other risks could be raised. A respondent at

GeoTech stressed that they treat the security department as an internal customer that provides requirements which are put on the backlog and prioritized by the product owner.

Security department

Most case companies have a dedicated security department. The tasks of this department often covered monitoring activities and high level security tasks that ensured that the teams would implement security properly in their products. For example, the security department at InsuranceInk handles security exceptions like temporary authorization and monitors whether the teams meet the requirements of the SDL framework. Furthermore, security officers provide advice and organize the execution of penetration tests. The security department at Govtech also conducts regular security awareness campaigns and encourages employees to think about security. Similarly, the respondent at FinTech stated that the company was currently busy developing security trainings for DevOps teams.

A respondent at GeoTech pointed out that the security department makes agreements with other departments so if his team request a server from the infrastructure team they can be assured that it is secure because of the agreements the infrateam made with security. Their security department also serves as a contact point in case the teams have questions. Multiple companies also involve their security departments in the design process of new systems or when choosing suppliers. At InsuranceInk the security manager is also involved in the steering committee of the DevOps transition.

In some companies the security department also conducted lower-level tasks. The security engineer at WebSales1 is responsible for auditing the teams from a security perspective and conducting PCI audits. He is also the contact person for advice and participates in CABs in case complex changes are submitted. The security department at WebSales2 conducts monitoring activities and reacts to threats e.g. by blocking traffic. The security departments at GeoTech equally conducts strong monitoring. At SmartIndustries, information security is covered by a group of people in the IT department who conduct internal audits and tests. They try to hack existing applications and are involved in designing new ones. There is also a cybersecurity business unit although the case study team is not involved with them very much. PublicServices also divided their security tasks across multipe departments. Security advisors think about infrastructure and appliances whereas security officers handle security policies such as ISO27001.

Outsourced platforms

Companies that had outsourced their platform to an external party stated that the hosting provider conducts extra monitoring activities. One respondent also stressed that they had service level agreements with their hosting provider that ensured security.

Security tests

Multiple tools were mentioned that aid the teams in developing secure products. Some teams at PublicServices use the security tool Fortify, however, one engineer of Team External noted that Fortify was not fast enough for his team. His team uses a static code analysis tool SonarQube but wants to move back towards Fortify in the future. The respondent at WebTech stated that one of their clients uses the open source OWASP scanner to check whether the microsites which WebTech produces for them are secure. GovTech and PublicServices hire third parties to conduct additional security testing like pentests in case they deem this necessary.

Most case study companies have only partly automated their security tests so far. Notable exceptions were WebSales1 who uses security testing software like pentesting software and specialized products for websites and webservices and WebSales2 who implemented automated security tests in their new way of working. Generally, companies automated more of the functional tests than security tests.

Security operations centers

PublicServices and SmartIndustries both had an internal security operations center (SOC). During a visit to the SOC at PublicServices, it was demonstrated that it did not only monitor logs but also monitored Twitter feeds that related to their work and associated instances so that the public opinion and external events could be detected quickly. The SOC also coordinates the Security Information and Event Monitoring (SIEM) which is partly conducted by an external party. Besides conducting monitoring, the SOC can also scan the infrastructure. At SmartIndustries, the SOC was located in another city and monitored running applications for which the case team requested this.

6.5.4 *Compliance*

In order to ensure and verify compliance, multiple companies had employed compliance officers or internal auditors. In the case of WebSales2 this was a risk & compliance officer whereas PublicServices had internal auditors. WebSales1 had employed a security officer who is partly responsible for security related audits. As the risk & compliance officer at WebSales2 pointed out, the company separates between critical applications that fall under the SOx regulations and less critical applications. Only for the SOx applications did the company use the COBIT framework. One team at PublicServices was building a so-called "audit robot" that would check in the logs associated to a committed piece of code whether it was first analyzed with the security tool Fortify. This would then allow the company to identify code that was potentially not sufficiently tested for security issues and call the responsible teams to account. Many companies are also regularly audited by third parties in terms of security, e.g. because they followed the ISO 27002 norm or use the identity management service DigID. None of the companies that already had some experiences with external audits such as security or financial statement audits reported particular difficulties due to their DevOps processes. Some respondents even mentioned that they were now able to handle concerns which the auditors mentioned faster due to working DevOps. SmartIndustries was the company with the strictest compliance requirements. One of the respondents mentioned that every time they want to implement a new application which is connected to the internet they have to let another party check this application.

Some controls found during the literature review in Table B.2 were also employed by the case study companies; however, this was not done because of specific compliance requirements but rather because companies viewed them as a good practice. All companies separated their testing and production environments from each other, mostly following the traditional DTAP pattern. Item tracking was often done in tools like JIRA by starting with epics and refining them to more specific user stories, tasks and sub-tasks. Controls that were implemented due to compliance requirements as described in Section 6.2.2 mostly related to change and access management. No company had automated compliance testing to monitor whether these compliance requirements were fulfilled. This was mainly done by internal auditors that were specialized in security or legal issues. The security manager at WebSales1 mentioned that, although general compliance checks are more difficult to deploy than security checks, it is e.g. possible to automatically check whether employees use suitable encryption algorithms that ensure data privacy. He also pointed out that it is in some cases still necessary to conduct manual checks especially when someone raises the suspicion that something is not right. It is therefore not effective to completely rely on the automatic compliance and security controls without critical thinking.

Another aspect that was often mentioned in order to ensure compliance requirements was to increase transparency and traceability of actions. This was mostly done by logging actions.

6.5.5 *Monitoring and logging*

While logging was acknowledged as important by all participants, the focus of logging varied slightly. While GovTech stressed that they conducted technical and functional logging, SmartIndustries stated to mainly focus on technical logging. According to the risk & compliance officer at WebSales2, strong monitoring is also a way to compensate for a potential lack of preventive controls of which they implemented less because they did not want to slow the process down:

> *"This is something very natural in DevOps that they [the developers] will also do the troubleshooting and deployment and everything. So they still have this access and the code can still change. So I think that would be one of the main remaining risks. It is not 100%. So usually companies would try to limit this level of access or put strong monitoring into that. Which is a way to compensate the missing piece of control there."*

The security engineer at WebSales1 pointed out that the DevOps teams are innately encouraged to monitor because they also have to provide operational services for their applications. Without the feedback they receive from monitoring these systems it is more difficult for them to detect problems. Encouraging the teams to do so is important because the teams are responsible for building the logging tools for their applications themselves, which create important audit trails. At GeoTech, at least one of the case teams uses the tool Splunk for monitoring operations. They have also created rule-based alarms, e.g. if the CPU is higher than 5%, an automated alarm is sent to the team mailbox. The administrator of a sprint is responsible for checking this mailbox as well as checking Splunk itself regularly to ensure that no other incidents happened which did not trigger rule-based alarms. Furthermore, multiple teams were observed to have large screens in their department that reflect real-time testing results and other important KPIs or logs. This was for example the case at GovTech and GeoTech. Seeing these test results continuously allows the teams to react fast to potential incidents and continuously improve their tests and code.

Besides holding the teams responsible for monitoring their own applications, many companies charged the security department with high-level monitoring tasks of network traffic or activities (also see Section 6.5.3). At some companies the security department also monitored internal processes: Every time a developer at GeoTech conducts an action that is not normally allowed for him, an alarm is sent to the security department. This also happens every time a developer deploys something into production. In case a change is registered and was previous authorized by the CAB, security conducts no further action. Otherwise, the notification is followed up. The security department at GeoTech also has dashboards with flags in their office in order to recognize incidents as fast as possible. At the companies that had outsourced their platform, the external hosting provider also conducts substantial monitoring activities. As mentioned earlier, two companies also had an inhouse security operations center which monitored logs and applications.

Various metrics were mentioned to be suitable for monitoring. The respondent at FinTech mentioned that the teams in her company get to decide themselves which KPIs they want to monitor. Examples for KPIs she gave were *mean time to restore service* or *change success rate*. An interviewee at InsuranceInk also stressed that it is important to conduct a zero measurement before the teams implement DevOps in order to track improvements through the use of DevOps. He also noted that his company currently uses old KPIs that should be changed to efficiently support the DevOps processes. FinTech also extracts monthly reports from their administration software ServiceNow that shows configuration management, incidents and changes. Similarly, SmartIndustries stated to regularly create reports for auditors but also to monitor their internal effectiveness.

Other controls that have been previously mentioned but are also important in the context of logging are change registrations, version control tools and logging access to production in order to trace who checked in, approved or deployed a piece of code. The respondent at Webtech also stressed the use of Docker containers in the context of logging: Containers do not only enable developers to update applications much easier but also allow them to trace application-versions more easily with a version number.

6.5.6 *Others*

Risk evaluation sessions

Many companies have risk evaluation sessions in which potential risks of a project or a new work item were discussed. WebSales2 calls this a threat analysis and conducts this during the design stage of every new project. In most companies, these meetings involve experts from different domains such as audit, legal, compliance and security but in some companies also customer centric disciplines like UX are involved. GovTech, FinTech and PublicServices do this on a regular basis or before every large project, similar to WebSales2. FinTech also mentioned to have additionally planned this for their DevOps implementation project. WebTech has also conducted a similiar workshop a while ago in order to identify security risks.

Risk log

SmartIndustries explicitly defines and monitors risks in large projects. GovTech stated that they incoporate risks into user stories which are documented in JIRA. WebTech also mentioned that they started being aware of potential risks some time ago but are not yet explicitly managing them or writing them down.

Rollout and rollback strategies

Most companies expect teams to have rollback plans in case a deployment does not perform as desired. How these are conducted varied per team. At InsuranceInk, teams are among others required to have a rollback plan in place when independently deploying quick releases that do not have to be approved by a CAB. GeoTech conducts rollbacks by making snapshots of the virtual servers and deploying the older version again. This has the advantage that no traces of the old, replaced application are left behind. The respondent stressed that this is only possible because they separate their databases from the applications which means that the databases are untouched if the applications are reset to an older version.

Team External at PublicServices uses a BlueGreen deployment strategy (see Section 4.5.6). The Microsoft Azure platform which they use enables the team to roll out versions to different regions like Northern Europe or Western Europe. Furthermore, they have a production and staging slot which they can swap in case the production version does not work as expected. The product owner of Team Internal mentioned that this strategy was not possible for teams using COBOL which only allows them to do rollforwards. These teams usually first check whether they can handle the issue in production and then deploy a better version. In case this is not possible, they have to roll out the old version again.

6.5.7 *Soft aspects*

As already evident, DevOps strongly emphasizes soft controls. During the interviews, various soft controls were identified that supported the implementation of above mentioned controls or mitigated risks identified in Section 6.3. Most of these controls were not mentioned explicitly by respondents but were identified from examples the respondents gave about how their teams behaved. While some of these aspects were already addressed in the previous sections, we will give a comprehensive overview of these controls in the following.

Clearly communicated responsibilities

Some companies demonstrated that they had clearly communicated responsibilities, authorizations and procedures in place. These clearly communicated responsibilities were found within the teams as well as when the team had to interact with people outside of the team: Within the teams, members knew very well what was expected from them in terms of development, maintenance but also security and compliance of their products. Multiple interviewees stressed that it was very important that teams are aware that they were end-to-end responsible for their product. As one respondent put it:

> *"Everyone has to be able to operate the run, if the system is in lock-down it could also be a developer who is called out of his bed."*

Outside of the teams, members also knew which person they should approach in case of incidents, questions or if a certain process needed to be authorized. Examples of these clearly communicated responsibilities include the appointment of data owners who need to authorize access to particular data sets and information security officers.

Team autonomy

In most companies, teams were given the freedom and authority to take independent decisions concerning their systems. If teams do not have the autonomy do take decisions and react quickly, this can increase security issues. Teams evidently need to receive this autonomy from their superiors which requires strong management support. Team autonomy is also strongly related to the concepts of organizational culture, continuous improvement and team responsibility. A manager at InsuranceInk illustrated this in the following example:

> *"I sometimes say as a joke that I would like to go back to the basics. Of course we are a corporation, but really I would like to go back to the basics in the sense that a team feels as if they were eight nerds in an attic. Like, you have your own company, your own responsibilities, that's clear, and the moment you don't have a product...well you have to put food on the table, the chimney has to smoke. Then you get this kind of spirit that teams really go for the very best. And that is were we want to go, really towards small, autonomous teams with a high degree of automation and efficiency."*

Communication

Teams were encouraged to communicate among each other as well as with managers at all times. This was mentioned very often in the context of change management when respondents gave examples of change deployments that also impacted systems of other teams (see Section 6.5.1). However, communication was also expected if teams had problems they could not solve themselves or had questions. One product owner mentioned that he consciously chooses not to participate in the daily stand-up meetings of his team but rather waits until the team approaches him with questions or problems. Teams should also feel responsible for their product and step forward in case of concerns as demonstrated in Section 6.4.6.

6.6 the devops transformation at geotech

The teams at GeoTech transitioned to DevOps one and a half years ago. Prior to this, they were working with an external supplier while GeoTech created the requirements and also internally operated the applications. However, changes involved an extremely high amount of documentation, multiple contracts and slow communication between all parties involved:

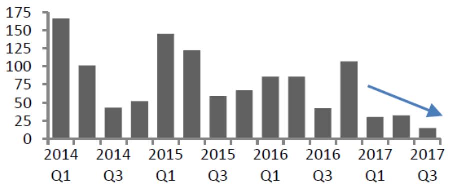
> *"It became completely unworkable with an external supplier and an internal operating department and another IT party in between. It went over too many levels with too much documentation. Of every change 80% was documentation of which only 20% finally became a change. Effectively we only did a few changes per three months, almost nothing."*

As part of the DevOps transition, the external supplier was transformed into consultants that worked on a hiring base and the functional and technical application management teams were merged and transformed into DevOps teams. All team members are now encouraged to perform both functional as well as technical tasks. The teams are responsible for the applications as well as maintaining the database behind their applications. Reaction times are now much faster because client requests are directed at the teams directly and are solved within the teams.

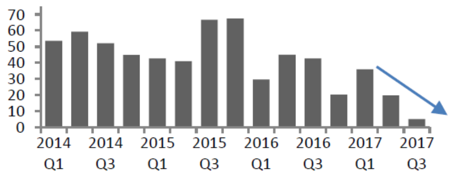
The DevOps teams have already automated some small tasks such as data extraction, transformation and loading. According to one respondent, the ultimate goal is to completely automate the process and to integrate all operational tasks into a CI/CD pipeline. Since the teams are responsible for both development and operations, they have administrator rights on all DTAP environments. However, team members do not have access to production by default but receive temporary access once a change is formally approved by the company which makes sure that changes do not clash with other scheduled changes. Small changes that are categorized as low impact changes can be performed directly by the teams which makes the process more efficient. All actions are logged and are therefore fully traceable. Version control is performed both on code and data as well as on configuration sets. Snapshots of the virtual servers enable the teams to perform quick and clean rollbacks. GeoTech also has a large security department which monitors the security logs. Security requirements from this department are treated like client requests and are added to the backlog and prioritized by the product owners.

GeoTech also has procedures in place in order to enforce communication and detect problems early. Team leaders have to inform their team members within an hour if they run into a problem which they haven't solved yet. If the problem is still not solved within a specified period, it is scaled up by informing the product owner, solution architect and other relevant actors. This way, serious problems are scaled up to an organizational level within less than a week.

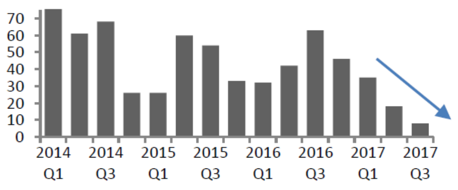
The transition has led to significant performance increases as shown in Figure 6.2. The represented team transitioned to DevOps in 2017 whereby the statistics clearly show decreased numbers of incidents and shortened resolution times within the three quarters after the transition. Most strikingly, the teams have managed to significantly increase their numbers of changes and to shorten the throughput time of these.



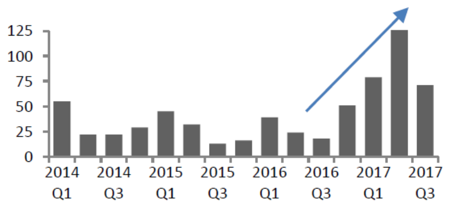
(a) Number of incidents



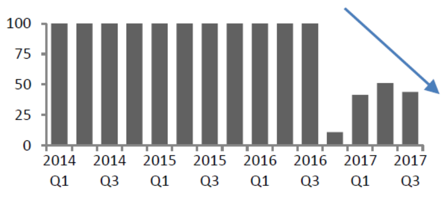
(b) Incident resolution time (days)



(c) Open number of incidents



(d) Number of changes



(e) Change resolution time (days)

Figure 6.2: Incident and change statistics GeoTech

7 THE DEVOPS RISK MANAGEMENT FRAMEWORK

The following chapter introduces the final risk management framework and strategies as designed after the validation interviews. For information on the draft framework and adjustments to the same refer to Chapter 8. The risk management framework introduced in this chapter consists of a classification of risks (Section 7.1.1), a summary of risk management practices that are compatible with DevOps (discussed in Section 7.1.2, complete overview in Appendix D), an overview of DevOps risk governance components (Section 7.2) and a risk management matrix that suggests four different risk management strategies for companies (Section 7.3).

7.1 SYNTHESIZING LITERATURE AND EMPIRICAL FINDINGS

7.1.1 *Risks*

As concluded in Chapter 4, risks were hardly mentioned in literature, although they have been identified plenty by practitioners. Furthermore, literature mainly focused on inherent risks that were directly related to the concept of DevOps (such as a high deployment rate or developers having access to production) while practitioners also mentioned practical risks that were related to the consequences of using DevOps e.g. planning and communication problems or teams becoming too autonomous. Moreover, literature mainly mentioned risks as a justification for suggesting specific controls instead of conducting thorough risk analyses.

Since the risks encountered in literature were much more generic than the risks mentioned by practitioners, it is not possible to compare these findings without identifying the root problems behind these generic risks. For example, the risk of frequent deployment in reality concerns the risk of deploying faulty products that do not meet security or compliance standards or lead to system downtime. This risk therefore falls into the product risk category as described in Section 6.3. Developers having access to production was a risk that was also mentioned by practitioners and was classified as a risk of internal fraud which again belongs to the team risks. The risks of security and compliance as mentioned in literature also concern the product that is deployed.

Analyzing the risks encountered in literature this way supports the classification of risks as designed during the case study analysis, whereby the risks mentioned in literature exclusively fall into the team and product risk categories. Figure 7.1 visualizes these risk categories and indicates a top-down approach that organizations can use when examining risks relevant to their situation.

The definitions of the categories adhere to the descriptions developed in Section 6.3 with the following additions: The *product* risk category does not only encompass the obvious products and services of an organization but more specifically the product the team is working on. This for example also includes the infrastructure and the CD pipeline since the infrastructure is nothing more

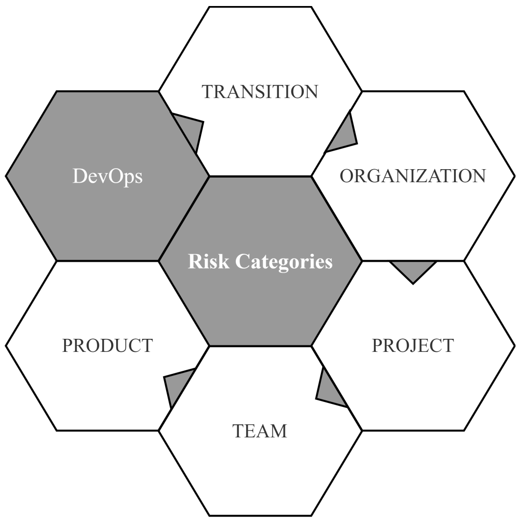


Figure 7.1: Risk categories related to DevOps

than a product maintained by the infrastructure teams while the pipeline either belongs to the product of the DevOps teams or is a separate product of a dedicated team maintaining the pipeline. Furthermore, the *project* risk category might seem counter-intuitive at first sight since projects are a traditional concept. However, the category describes any activity that is bound to a schedule and budget and involves multiple actors. This can refer to development activities within a team or to organizational wide activities. Because many companies still use this term in combination with the described activities, it was decided to keep this category with the given name.

Naturally, these categories are interconnected since a problem in one category can lead to risks in another category. Furthermore, some categories might overlap as one risk could be assigned to multiple categories. Nevertheless, the overview is deemed to cover all important risks and can therefore be a useful tool which aids organizations in conducting a complete risk assessment for their DevOps processes.

7.1.2 *Risk management practices*

A comparison of risk management practices in literature and practice shows that literature suggests many technically advanced, automated controls while companies in practice still rely on more traditional practices because they are not yet mature enough in their processes to rely on the other controls. Most controls in literature were also recognized in practice, although the technically advanced controls were only found in very mature companies. Furthermore, controls from literature were mainly focused around the operational processes regarding software delivery and were less concerned with supporting mechanisms such as the role of the security department.

A complete overview of all practices found in literature and practice can be found in Table D.1 in Appendix D

While literature suggests compliance as a goal in itself, respondents from the case studies made clear that compliance for them is not a goal that needs to be achieved but rather viewed it as a basic condition they need to fulfill from where they design their operations. Furthermore, many compliance controls that were mentioned usually fall into one of the other categories such as security controls (when following security frameworks) or change and access controls (e.g. when applying ITIL or COBIT). Nevertheless, we will keep compliance as a separate category in our framework to cover controls that ensure general compliance and controls that may be required in terms of compliance and not fall into one of the other categories.

7.1.3 *Mapping controls to risks*

Table 7.1 summarizes the identified risks and suggests some controls that can be applied to mitigate these. Besides mapping controls found in literature or during the case studies to the risks, some controls had to be newly designed or specified more in depth to cover the risks sufficiently. This mainly concerns controls about awareness trainings as well as the specifications of soft controls and general risk mitigation mechanisms. As evident, some controls were also used in a different context than originally specified. For example, the control *reporting* is not only useful for reporting to auditors but can also be useful as a monitoring tool in which DevOps teams have to issue these reports to the management in order to demonstrate and justify their actions and choices.

The table also clearly demonstrates the importance of soft aspects such as *communication* and *culture*. Since soft aspects in DevOps are no longer just useful for improving business processes but are rather integral controls to mitigate important business risks, it is essential for companies to create more measurable and verifiable approaches to soft controls. Being able to demonstrate the effectiveness of their soft controls will also allow companies to prove their internal control to auditing parties. For example, *culture* could be demonstrated through the existence of a code of conduct or by conducting regular trainings while *communication* could be demonstrated through the establishment of effective communication channels and communication protocols.

Table 7.1: Examples of risks and controls

This table only intends to provide examples of important risks and suitable controls to mitigate these. It does not aim to be complete

CATEGORY	RISK	CONTROL
Transition	1.1 Teams don't understand and carry out their responsibilities which affects security and quality of work	1.1.1 Regular awareness trainings 1.1.2 Clearly define and communicate team responsibilities 1.1.3 Regular audits of teams and code by relevant departments (e.g. security, compliance)
	1.2 Stressed or frustrated employees lead to less productive work	1.2.1 Clearly define and communicate team responsibilities 1.2.2 Appoint contact people outside of teams for questions and support (e.g. security and compliance departments)

CATEGORY	RISK	CONTROL
Organization	2.1 Collaboration of DevOps and non-DevOps teams leads to scheduling and resource problems	2.1.1 Communication 2.1.2 Apply some project management principles to ensure proper time and resource planing 2.1.3 Limit contact between DevOps and non-DevOps teams 2.1.4 Team autonomy
	2.2 Management and DevOps teams have differing priorities which leads to scheduling and resource problems	2.2.1 Management support 2.2.2 Communication 2.2.3 Team autonomy
Project	3.1 DevOps teams don't meet customer deadlines because of Agile working or unforeseen incidents	3.1.1 Use agile frameworks that help to structure the process (e.g. SAFe) 3.1.2 Use some traditional project management mechanisms 3.1.3 Train product owner to prioritize backlog items and incidents
	3.2 Project fails because of too little overview	3.2.1 Use some traditional project management mechanisms (e.g. quality gates, project board, risk officer) 3.2.2 Clearly define and communicate responsibilities
Team	4.1 Wrong prioritization of work items leads to critical items not being carried out	4.1.1 Train product owner to prioritize backlog items and incidents 4.1.2 Clearly communicate responsibilities
	4.2 Team member conducts internal fraud by stealing data or manipulating systems	4.2.1 All Identity and access management controls 4.2.2 All change controls 4.2.3 Logging 4.2.4 Monitoring for unauthorized actions 4.2.5 Soft controls (e.g. team responsibility, non-disclosure agreements)

CATEGORY	RISK	CONTROL
	4.3 Teams become too independent and their actions cannot be controlled by the organization anymore	4.3.1 Logging 4.3.2 Communication 4.3.3 Soft controls (e.g. code of conduct, responsibility trainings, culture) 4.3.4 Reporting
	4.4 Teams don't stick to schedule that was agreed upon which jeopardizes on-time delivery of work	4.4.1 Soft controls (e.g. code of conduct, responsibility trainings, culture) 4.4.2 Clearly communicate responsibilities 4.4.3 Reporting
Product	5.1 Deployed product is not secure	5.1.1 All security controls 5.1.2 Monitoring network traffic 5.1.3 All change controls 5.1.4 Soft controls (e.g. team responsibility)
	5.2 Deployed product is not compliant	5.2.1 All compliance controls 5.2.2 All change controls
	5.3 System malfunctions because of faulty product	5.3.1 All change controls 5.3.2 Rollback strategies
	5.4 Deployed product is not of sufficient quality	5.4.1 Automated functional tests 5.4.2 Quality Assurance practices (e.g. QA Officer, regular quality audits)

7.2 DEVOPS RISK GOVERNANCE COMPONENTS

The main elements that constitute effective DevOps risk governance are visualized in Figure 7.2 in the form of a house. The roof represents the risk categories and compliance requirements that apply to a company and which should govern all organizational design choices. Clearly assigned roles, responsibilities and policies are the overarching requirements for effectively dealing with these risks and compliance requirements. These general mechanisms are supported by four pillars: The first three pillars represent the three lines of defence model (which was previously described in Section 2.2.3) in a DevOps context. The first line addresses operational processes around software delivery and operations and contains the automated and manual controls. The DevOps teams themselves are the risk owners and are responsible for the implementation and execution of the first line controls. The second line contains supporting and overseeing functions

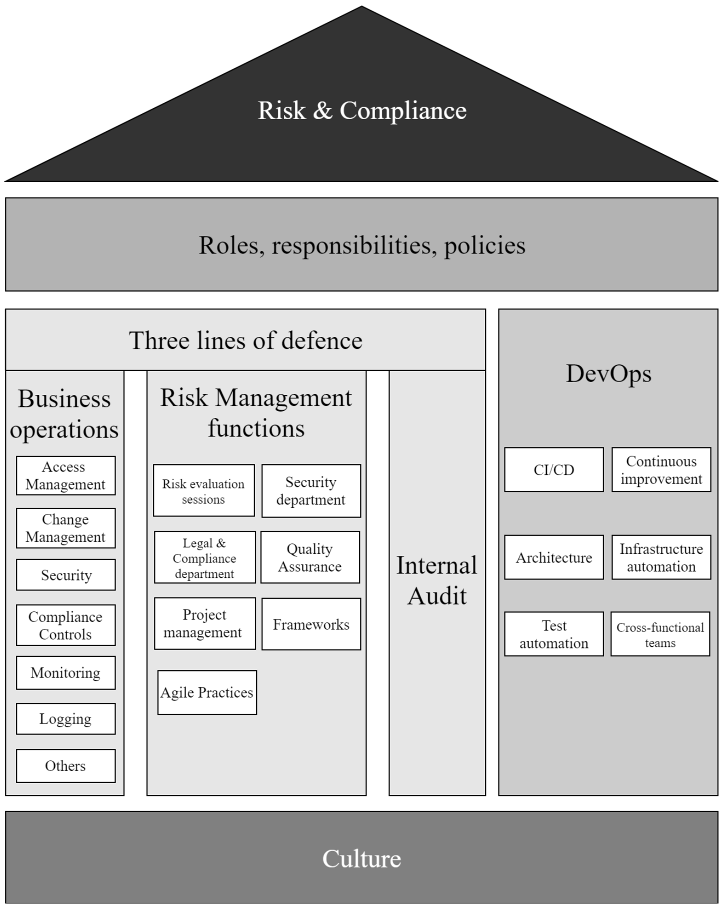


Figure 7.2: Representation of DevOps risk governance components

that actively manage the risks. This includes departments that have been established as important for supporting the DevOps teams as well as frameworks and methods they provide. Besides the security, legal and QA departments mentioned in this figure, other departments may have to be involved in the DevOps process depending on the exact activities of the teams. The last line of defence provides assurance in form of internal audit. Although not all case study organizations conducted explicit internal audits, we have incorporated this practice to represent the necessary recurring assessment of the efficiency of risk management activities in the model. The DevOps processes add business value and operational efficiency. Some DevOps processes naturally overlap with the proposed controls. DevOps can e.g. contribute to effective monitoring, logging, conducting quality assurance through automated testing and conducting version control. The continuous improvement aspect is also reflected in the internal audit and the culture. A strong DevOps culture emphasizing integrity, trust and communication is the foundation of all other components in the house, clearly demonstrating that a lack of culture will lead to a collapse of the DevOps organization.

The blocks are based on the concept map in Figure 6.1. However, some elements have been rearranged due to their priorities. The case studies and literature review have made clear that culture is one of the most important elements in DevOps risk management and cannot simply be seen as an interchangeable risk management process. It has therefore been moved towards the bottom of the house. The identified soft controls match with the elements *Culture* and *Roles, responsibilities, policies* and are therefore not mentioned separately in the operations pillar anymore. Lastly, the product owner who was identified as one of the main actors in mitigating product and team risks is also incorporated in *Roles, responsibilities, policies* and in *Agile practices* which represents the importance of his function better.

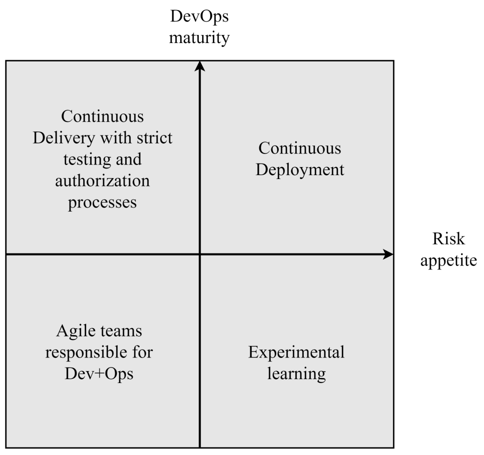
While this figure represents all important elements of risk governance, it does not give information about the design of the risk management strategy that companies should drive nor the specific practices which it should implement. During the literature review, many controls and mechanisms have been identified; however, not all of them are suitable for every situation. The following sections will address this issue and aim at establishing guidelines for how the elements and processes represented in this house should be designed and executed.

7.3 THE DEVOPS RISK MANAGEMENT MATRIX (DRMM)

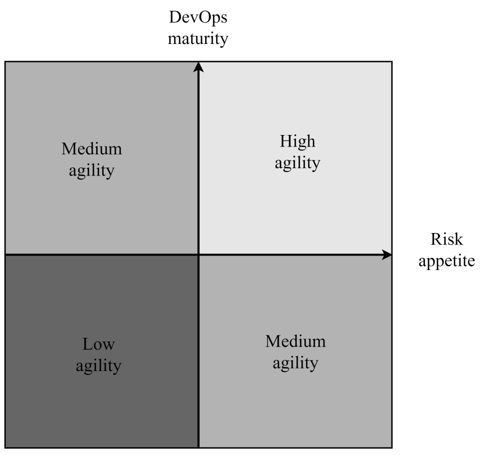
As identified in Section 6.2, two main factors which predominantly influence the nature of risk management practices are the *compliance requirements* to which a company is subdued as well as the *DevOps practices* which it implemented. Extending these concepts to broader categories which they represent leads to the proposition that two main factors which influence companies in the design of their controls are *Risk appetite* as well as *DevOps maturity*. Risk appetite is based on the impact that the occurrence of a risk will have on the company and on the teams' choice whether it wants to carry these consequences. A higher risk impact as well as many compliance requirements often lead to a lower risk appetite: While e-commerce companies like WebSales1 and 2 chose to value the speed of deployment over not making smaller mistakes such as bugs, the same choice was not reasonable for companies operating in high-risk or high-compliance environments like FinTech and SmartIndustries. However, risk appetite was also found to vary heavily per team. Teams that were providing internal services to their company could usually afford to make more mistakes than teams providing services to customers.

Mapping these two factors on a matrix will give us the figure represented in Figure 7.3a with four basic situations in which teams can find themselves. We will call this matrix the *DevOps risk management matrix (DRMM)* in the following. The figure includes a basic description of a risk mitigation strategy that is suitable for each situation. While companies cannot completely control their position on the risk appetite dimension, they can change their DevOps maturity. The background colours in the matrix represent the agility which teams can achieve at each stage of their DevOps journey while remaining in control: Light background colours represent high agility and dark colours low agility (see Figure 7.3b). This division also explains the main driver why teams want to become more mature in their processes: Increasing DevOps maturity leads to an increase in agility and lets them reap more benefits of DevOps (some of which which have been identified in Section 6.2.4 earlier). While teams with a higher risk appetite and a high maturity can fully automate their process, transfer most of the responsibility to the teams and thus achieve the maximum possible speed and agility, the same degree of automation is not suggested for teams with a

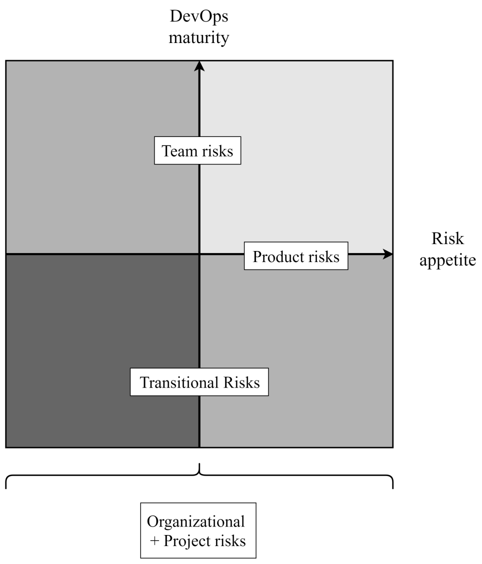
A detailed description of each DRMM strategy is given in Section 7.3.2


DevOps maturity
Continuous Delivery with strict testing and authorization processes
Continuous Deployment
Risk appetite
Agile teams responsible for Dev+Ops
Experimental learning

(a) Basic description of DRMM strategies



(b) Maximum speed that can be achieved while remaining in control



(c) Risks applicable to DRMM quadrants

Figure 7.3: Demonstration of DevOps risk management matrix

low risk appetite, even if they are very mature in their processes. This is due to the need of extra manual preventive controls that ensure compliance and risk mitigation before deployment.

While risk appetite and DevOps maturity seem like two different factors, it is important to note that the two axes are not entirely independent from each other. Teams that operated in a lower risk environment (and thus often had a higher risk appetite) were in general more mature in their DevOps processes than teams in higher risk environments. It is therefore likely more difficult for teams who are less capable of taking some risks to move from a low maturity of DevOps processes to the high maturity quadrant. This is likely because teams with a low risk appetite somewhat challenge the idea of DevOps as suggested by Farroha and Farroha [13] who claim that it is more important to recover quickly from mistakes instead of completely preventing them from happening.

7.3.1 *Relationship risks and DevOps risk management matrix*

After examining the basic risk-mitigation strategies demonstrated in Figure 7.3a it becomes clear that the risks identified in Section 6.3 have varying priorities in the different quadrants. While companies should of course always consider all types of risks and identify whether they apply to their specific situation, it is evident that some risks like transitional risks only apply to companies with a low DevOps maturity while others like team risks are mainly due to a high degree of team autonomy which is a characteristic of higher DevOps maturity. Product risks are generally higher in teams with a high risk appetite since these may employ less checks before deploying a product and prioritize speed and ultimately continuous delivery over (calculated) risks. Especially teams that deploy relatively often but have not yet implemented many functional tests are at risk of deploying faulty products. Organizational and project risks were not found to be related to risk impact or DevOps maturity but depend on the nature of the organization and the type of work that the teams conduct.

Companies have to pay special attention to implementing extra mitigating controls that arise from the risks mentioned in their teams' quadrant. These controls will therefore be integrated in the strategies that are introduced in the next section. It is important to note that companies which still have a low DevOps maturity and employ some traditional development methods naturally still have to be aware of risks specific to traditional software development which are not discussed in this thesis.

7.3.2 *The DRMM strategies*

As shown in Table 7.1, multiple controls can be used to mitigate the same risks. The controls a company chooses should therefore depend on its position in the DRMM and the corresponding strategy it should pursue, as well as on the risks that are of particular importance to the team's situation. Figure 7.4 suggests some controls that support the respective strategies. The strategies are discussed in more detail in the following.

High DevOps maturity - low risk appetite: Continuous delivery

Teams in the low risk appetite half of the matrix have to focus much more on designing preventive controls than teams in the other half because they can afford fewer mistakes than the prior group. Nevertheless, teams with a high DevOps maturity can automate much of their process and leverage DevOps to their benefit. Enforcing critical functional and security tests before every deployment can increase product quality and safety because it eliminates the possibility of human mistakes. On the other hand, in order to eliminate the possibility of technological malfunctioning or not testing important parts of a new feature, manual approval should be always required right before deployment. This can for example be done by the product owner of the team who verifies that the user requirements are fulfilled or through another DevOps engineer who can run some technical tests that verify safety and compliance. The exact design of these controls depends on the company's wishes and its particular circumstances. Large changes should always be approved by a CAB. Generally, developers

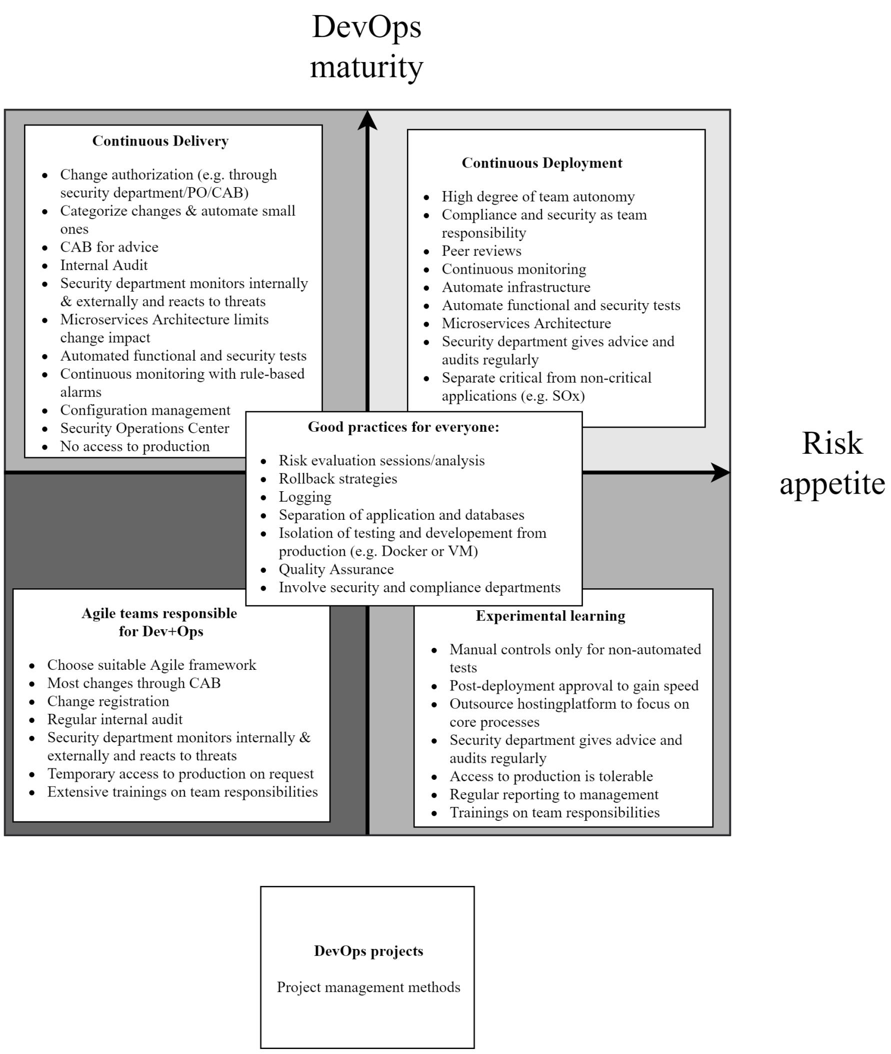

DevOps maturity
Risk appetite
Continuous Delivery
Change authorization (e.g. through security department/PO/CAB)
Categorize changes & automate small ones
CAB for advice
Internal Audit
Security department monitors internally & externally and reacts to threats
Microservices Architecture limits change impact
Automated functional and security tests
Continuous monitoring with rule-based alarms
Configuration management
Security Operations Center
No access to production
Continuous Deployment
High degree of team autonomy
Compliance and security as team responsibility
Peer reviews
Continuous monitoring
Automate infrastructure
Automate functional and security tests
Microservices Architecture
Security department gives advice and audits regularly
Separate critical from non-critical applications (e.g. SOx)
Good practices for everyone:
Risk evaluation sessions/analysis
Rollback strategies
Logging
Separation of application and databases
Isolation of testing and developement from production (e.g. Docker or VM)
Quality Assurance
Involve security and compliance departments
Agile teams responsible for Dev+Ops
Choose suitable Agile framework
Most changes through CAB
Change registration
Regular internal audit
Security department monitors internally & externally and reacts to threats
Temporary access to production on request
Extensive trainings on team responsibilities
Experimental learning
Manual controls only for non-automated tests
Post-deployment approval to gain speed
Outsource hostingplatform to focus on core processes
Security department gives advice and audits regularly
Access to production is tolerable
Regular reporting to management
Trainings on team responsibilities
DevOps projects
Project management methods

Figure 7.4: Controls for DRMM strategies

should not gain access to production. Furthermore, teams need to be responsible enough to reach out to the security department and the CAB in case they have questions or concerns instead of just pushing critical changes to production. The security department should furthermore take a strong monitoring role in this strategy, both internally and externally. This means that on one hand it should support the teams with monitoring their systems and infrastructure and by taking appropriate action such as blocking traffic if required. On the other hand, security should also monitor for internal incidents such as unauthorized actions by developers. These monitoring activities can be supported by triggering rule based alarms in case of predefined events. In very high risk environments that require extensive monitoring it is recommended to establish a dedicated security operations center either in-house or by contracting a third party that conducts strong and continuous monitoring activities. The company should also conduct regular internal audits to ensure the controls are still up-to-date and effective. Despite the large amount of preventive and detective controls, it is very important for these companies to train the team responsibility and awareness as well as to establish a culture of continuous improvement in which team members dare to speak up in case of concerns. Only if such a culture is created the company can trust their operational DevOps processes. On a general note, since approval in continuous delivery can be given with one single click on the button, this strategy does not loose much of its speed compared to continuous deployment, if the responsible person authorizes important changes right away.

High DevOps maturity - high risk appetite: Continuous deployment

This quadrant is undoubtedly the most desirable for teams to be in because it allows them to achieve the maximum possible speed and agility and lets them automate as many manual tasks as possible including the deployment of the product. However, this strategy is only responsible if the consequences of deploying a partially faulty product or a team not carrying out its responsibilities are relatively low and the company is willing to carry them. Companies operating this strategy can give their teams great autonomy and responsibility and can emphasize the use of detective controls (monitoring) over preventive controls. Teams are mainly responsible for security and compliance related issues themselves which increases speed and eliminates unnecessary bureaucratic processes. Since the deployment is fully automated, it is not necessary to give all developers access to production which lowers the risk of internal fraud. Furthermore, frequently deploying small releases also lower risk to some extent and lets these teams detect problems early. The security department takes an advisory role which the teams can approach if they have questions and audits the processes in regular intervals. In order to maintain a separation of duties principle and prevent internal fraud, peer reviews for approving merge requests before entering the deployment pipeline are recommended. However, in order to compensate for the lack of preventive controls, teams should always have a rollback strategy in place and conduct strong monitoring activities that detect and escalate incidents quickly. Teams in this quadrant can also consider making use of platforms such as Microsoft Azure or AWS which do not inhibit their continuous deployment capabilities.

Low DevOps maturity - high risk appetite: Experimental learning

The third quadrant describes companies or teams that are willing to take some calculated risks but are not yet very mature in their processes and actions. Due to their willingness to accept the impact of small mistakes, teams in this quadrant are free to experiment with different controls and techniques to find out what works best for their situation. Teams in this half of the matrix typically prioritize speed over perfection which is why controls such as post-deployment approval are acceptable here. Obviously, once a company matures in their processes it can still implement more automated tests which don't take away any of the speed but lowers the possibility of product risk (see Section 7.3.2). The team might also benefit from BlueGreen deployment strategies and A/B testing to test customer responses to different versions of their product. Since the risk impact is presumably lower for teams with a high risk appetite, it is generally acceptable (although not desirable) to give developers access to production e.g. to troubleshoot, although companies will then have to implement strong monitoring and logging instruments to ensure traceability. If the teams want to focus on building their core capabilities and focus less on infrastructure, they can also benefit from outsourcing their platform to a third party or using cloud services such as Microsoft Azure or AWS. Nevertheless, teams should conduct thorough risk analyses and not refrain from using manual controls to compensate for missing automated tests, even though these might inhibit their agility to some extent. Since the teams are still in the starting phase of their DevOps transition and might not be accustomed to their responsibilities and the DevOps culture yet, management should review their actions regularly, for example by asking for updates and reports from the teams. Teams should also receive extensive coaching on their responsibilities to prepare them for working more independently once they reach a higher DevOps maturity.

Low DevOps maturity - low risk appetite: Agile teams responsible for Dev and Ops

Teams that are not yet mature in their DevOps processes but cannot or do not want to take many risks are operating in a rather perilous environment. They therefore have to be extremely careful when implementing new features and methods. A suitable starting point is to implement an Agile framework like Scum or SAFe that structures their processes and to give teams responsibility for both development and operations. Automation does not yet play a (big) role at this point because it is more important for teams to become accustomed to the DevOps culture and their broadened responsibilities. Controls are therefore rather traditional and can be replaced by automated controls during the growth process (see Section 7.3.3). Changes should generally be approved by a change advisory board, except for very small changes with minimal impact, and should always be registered beforehand to improve traceability and inform other teams. Teams however should be extensively trained in their responsibilities in order to prepare them for a more autonomous way of working. Developers should only gain temporary access to production in order to deploy changes, e.g. by requesting timed passwords. In order to assess and ensure compliance, the company should also employ internal auditors or risk & compliance officers.

7.3.3 *How to move from one quadrant to another*

The risk appetite of a company or team will likely not change through the implementation of DevOps which means that teams are operating on a vertical scale in the DRMM, aiming to achieve a higher DevOps maturity. The main drivers for this desired maturity growth has been identified above as increased agility and generally reaping more benefits associated with the DevOps concept. As mentioned earlier, it is easier for teams with a higher risk appetite to move to a higher maturity than for teams with low risk appetite.

Figure 7.5 gives a basic indication how teams should approach their maturity growth process based on the available strategies. While teams with a higher risk appetite are rather free to experiment with different deployment strategies and automated controls, teams that don't want make mistakes have to think much more about the applicable risks and mitigating controls beforehand. The risk assessment approach suggested in Figure 7.1 and the complete summary of controls in Appendix D can be of help when doing so. Teams in this situation also need to identify and involve critical actors in the design of their DevOps processes from an early stage on. Most importantly, this concerns the security department but could also apply to departments such as quality assurance, infrastructure and legal & compliance. While this overview gives a basic indication on achieving maturity growth in DevOps, more research on DevOps maturity stages and growth strategies needs to be conducted to give more specific instructions.

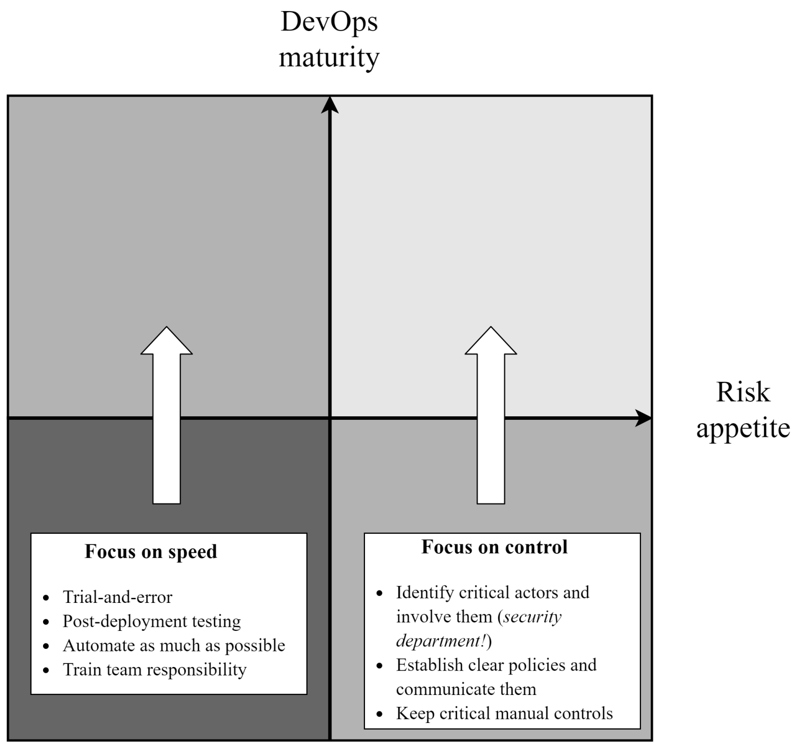

DevOps maturity
Risk appetite
Focus on speed
Trial-and-error
Post-deployment testing
Automate as much as possible
Train team responsibility
Focus on control
Identify critical actors and involve them (*security department!*)
Establish clear policies and communicate them
Keep critical manual controls

Figure 7.5: Basic growth strategies

8 VALIDATION

This chapter summarizes the validation interviews and the results of the validation survey and the adjustments that were made to the draft framework based on the received feedback. As shown in Table 8.1, four experts working in the fields of DevOps and/or IT Risk and audit were interviewed during this phase. The interviews are summarized in the following sections. Finally, Section 8.5 summarizes the results of the survey sent to the case study respondents.

Table 8.1: Experts interviewed for artifact validation

POSITION	EXPERTISE	EXPERIENCE
Senior Manager - GRC Technology	Governance, Risk & Compliance, IT Audit, DevOps	9 years
Senior Manager - Digital Enablement	Agile, DevOps	11 years
Senior Consultant - Enterprise Agility	Agile, DevOps, Change Management	10 years
Director - IT Assurance & Advisory	IT Audit, IT Risk	12 years

8.1 SENIOR MANAGER - GRC TECHNOLOGY

The first expert to be interviewed was a senior manager in the Governance, Risk & Compliance Technology department of a Big4 audit- and consultancy company. The interviewee had almost nine years of experience in IT risk consulting as well as IT audit and IT assurance engagements and is currently also consulting some companies on how to keep control over their DevOps processes. The respondent said that he experienced a shift of DevOps from more experimental web applications towards critical core systems over the past years which makes DevOps also a relevant topic for auditors nowadays. Nevertheless, he stated that his department is still exploring how to control DevOps and that not everything is completely clear to them yet. Since the respondent had extensive knowledge about both DevOps and IT Risk, the interview served to receive broad feedback on the whole research and to improve the framework based on this.

8.1.1 *Risks in DevOps*

The expert thought that risks can be viewed from many different angles: While auditors during the financial statement audit will look at the integrity of the systems processing the data, other others might look at to effectiveness, privacy and continuity. Looking at the classification of risks as done in this research, he noted that many of the categories were interconnected and that many risks such as problems in the organization or transition would eventually lead to product risks. Generally, he thought that the categories represented different levels in which the risks would manifest themselves but if one looked at the root problems, the categories did make sense to him. Furthermore, he also thought that one risk could fall into multiple categories. While the exact category of a risk is therefore not always clear, he agreed that the combination of categories did cover all risks he could think about. On a side note, he mentioned that the term *"project"* represented a rather traditional concept which implies that a product is specified, designed and finished at some point.

The interviewee also thought that it was important to stress that not all specific risks that were listed in Table 7.1 belonged exclusively to DevOps. For example, the risk of a team member conducting internal fraud applies to all companies and not only to DevOps teams. Nevertheless, the risk is obviously also important to consider for companies using DevOps. While the risk did not change by implementing DevOps, he noted that the controls to mitigate this risk did change in nature. He also pointed out that the DevOps practices such as faster and more frequent deployment brought new risks to the process. These were mainly classified as product risks.

8.1.2 *DevOps risk governance*

The first versions of the risk governance house consisted of three pillars being *control processes*, *DevOps* and *risk management practices*, similar to the concept map introduced in Figure 6.1. However, the interviewee thought that the division between controls and risk mitigation mechanisms was not entirely clear since they seemed to represent similar concepts. Furthermore, he noted that the use of frameworks, which were classified as a risk management concept, also influences other factors such as the roles, responsibilities and policies in the organization. Nevertheless, he overall agreed with the approach which the framework represented.

8.1.3 *DevOps risk management and controls*

The draft risk management matrix originally contained the axis *"risk impact"* instead of *"risk appetite"*. The respondent thought that these were two very suitable factors to use for the matrix. However, in order to make the meaning of the risk impact axis clearer, he proposed to rename the same to *"risk appetite"*. The main reason for this was that the respondent argued that a company can choose its position on the horizontal axis to some extent itself, by deciding whether it wants to carry the consequences of a risk occurring or not. This contradicted the original assumption made during the design of the draft framework that companies cannot influence their position on the risk impact dimension since this purely depended on external factors. Renaming the *"risk impact"* axis to *"risk appetite"* also implied that the matrix had to be mirrored vertically. The interviewee also agreed that the two axes are not entirely independent from each other and wondered whether a company in the high risk impact/high DevOps maturity segment even existed. He agreed with the strategies presented but did not recognize the difference between the lower two quadrants at first. The low maturity/high risk appetite quadrant was therefore renamed from *"Manual Controls only for non-Automated Tests"* to *"Experimental Learning"* to stress the experimental part of the strategy as opposed to the low maturity/low risk appetite strategy which was more controlled.

8.2 senior manager - digital enablement

The second interviewee was a senior manager in the Digital Enablement department of the same company as the first interviewee. He had worked for the company for eleven and a half years and has extensive experience with software development teams. In the past he has worked on software testing, software design and as a team lead for development teams. Due to the interviewee's specialization in agile methods and DevOps and his technical knowledge, the validation interview focused on the technical aspects of the proposed solution as well as how they could contribute to managing risks. The interview also addressed DevOps implementations on a general level and suitable strategies for conducting these.

8.2.1 *Risks in DevOps*

The expert agreed with the presented categorization of risks and thought that these covered all important aspects. He also agreed with the statement of the first interviewee that the risks were overlapping since e.g. problems within a project could affect the team and ultimately the product. He stressed that there would always be some overlap between the categories because one problem often causes another one. However, he thought that this didn't matter as long as the framework helps companies to identify the relevant risks. He particularly liked the mentioning of transitional risks which he deemed to be very important. He furthermore pointed out that infrastructure and the CD pipeline (which fall into the product category) were important elements to consider when assessing risks. He agreed with the first respondent that the term *"project"* is a rather traditional concept and that in DevOps this concept was often called "release trains" or "value streams". However, he also acknowledged that many companies were still using the word *"project"* and thought that this was rather a semantic discussion. On a general note, he thought that the risks in DevOps were not very different to risks in traditional software development although the controls did change and engineers now rely much more on automation. This supported the statement made by the first interviewee.

8.2.2 *DevOps risk governance*

The expert confirmed that *"roles, responsibilities and procedures"* were an important element when asked about this. He stressed that it was very important to have clear agreements in order to operate effectively as a team. However, he also noted that team members should have these roles and responsibilities internalized instead of just documenting them. When asked about the division between the *risk mitigation mechanisms* and *controls* pillars, he answered that he understands why the first respondent thought that this was confusing but also sees why they were set apart in the model since they were more of a basis or a starting point for companies to enforce effective risk management. Generally, the risk mitigation mechanisms were less tangible than the controls.

8.2.3 *DevOps risk management and controls*

The respondent agreed with renaming the *"risk impact"* axis to *"risk appetite"* and added that the concepts were related since the risk appetite of a company is heavily influenced by the impact of a risk on the company. He also agreed with the general strategies designed and pointed out that the low risk appetite/low maturity quadrant represented more of a Scrum way of working than a real DevOps team. However, he thought that the most important element was the culture of combining development and operations which was also included in this strategy. He also agreed with the mapping of risks to the matrix. Looking at the controls suggested for each strategy he agreed with the structure and pointed out that the controls in the lower half were more traditional while the ones belonging to higher maturity strategies were rather automated. He also stressed that automating the infrastructure e.g. by using containerization was also a kind of risk management control because teams would generally not have to access the product in the CD pipeline or production anymore. Looking at the risks and controls table, the expert added that pair programming, as for example done in Extreme Programming (XP), is also a method to ensure peer reviews.

The expert agreed that the role of auditors will change in the near future through the implementation of DevOps: whereas auditors in the past have sought assurance in documents and procedures, they will now have to look at toolstacks and logs to obtain the necessary information about automated controls. He also mentioned that he was surprised by the little amount of research that was done on DevOps until now, although many companies were already using it in practice.

8.3 SENIOR CONSULTANT - ENTERPRISE AGILITY

The interviewee was a senior consultant in the enterprise agility line of a large technology company. This line specializes in transformations towards Agile, including Lean, DevOps, SAFe, LeSS and similar principles. The interviewee has worked at the company for about 10 years and worked in healthcare before this. Due to this background he had a specialization in human and organizational behavior and change management which according to him was the most important factor in Agile transformations. The interview therefore had a particular focus on the human and cultural aspects of the proposed strategies and their expected implications.

8.3.1 *Risks in DevOps*

Concerning the overview of risks he added that quality risks were also important to consider. These were considered a valuable addition and were added to the product risk category. The respondent did not agree with the project risk category because he thought that organizations using DevOps should not conduct projects anymore. According to him, the main risk was that organizations were still thinking in projects which hinders agility. Nevertheless, he thought that it was recognizable that many companies were still using projects although they were working DevOps but said that this category will ultimately not exist anymore once the organizations grow in maturity and stop doing so.

8.3.2 *DevOps risk governance*

The respondent agreed with the DevOps risk governance overview. While he thought that some of the names given to the elements in the house sounded somewhat traditional, he agreed with the description he was given about these. He proposed moving the culture block from the roof of the house to the bottom to symbolize that it was the most important element and that the house would collapse without the foundation. Similar to the first respondent, he had some difficulties seeing the difference between the general risk management practices and control processes.

8.3.3 *DevOps risk management and controls*

Generally, the interviewee felt that many of the suggested controls were still too traditional and would only describe the first steps of a DevOps transformation instead of the ultimate goal. He did however fully agree with the suggested continuous deployment strategy since this did not make use of traditional controls. The interviewee was convinced that companies should also consider completely automating their delivery pipeline if they were operating in a high risk environment because he thought that people would make more mistakes than technology. The concept of continuous delivery was therefore redundant to him because it should not be necessary that a person authorized deployment if the company had already ensured that the scripts and tests were of sufficient quality. Again, he noted that it was logical that companies at this point in time still had the desire to use manual controls but that this should only be a first step and not the ultimate goal.

For the same reason the respondent did not agree with the project risk category, he did also not agree with the project management controls. Generally, he thought that it would be rather inefficient to try and create a mixture of PRINCE2 and Agile management methodologies and said that transitioning to DevOps required a completely new way of working that requires the company to drop old habits. In this context he also pointed out that employees transitioning to DevOps have to learn to trust the technologies instead of wanting to add manual checks that are not necessary.

The expert stated that he did not particularly like the term *"DevOps maturity"* because he did not like the idea of maturity models. However, since the term was used on the matrix he thought that users should receive some instructions on how to assess their DevOps maturity and risk impact since the advice they receive from the framework depended on their position on the matrix. This problem was already partially addressed by renaming the *"risk impact"* axis to *"risk appetite"* which demonstrates that this is a choice that should be made by the company. Furthermore, our framework does not aim to prescribe the exact controls a company should implement depending on their position on the matrix but rather show them possibilities which controls they can use depending on whether their teams are independent enough and have automated parts of their delivery pipeline. The choice whether a company should pursue a high or low DevOps maturity strategy is therefore also up to the company. The company will however be limited naturally in its choice of controls if it has not automated their pipeline yet or if teams are not yet trained enough to take on extra responsibilities.

8.4 director - it assurance & advisory

The last interviewee was a director of the IT Assurance & Advisory department in the same company as the first two respondents. He was specialized in IT assurance and audit engagements as well as IT Risk advisory projects and had worked in the field for over 12 years. The interview was therefore focused on how the different strategies could be assessed from an auditing perspective and whether the controls mitigated the risks sufficiently.

8.4.1 *DevOps audit and controls*

Similar to most other respondents, the expert acknowledged that the DevOps transition will lead to new ways of working for IT auditors. Instead of auditing the change process and particular objects, auditors will have to audit the automated scripts that ensure the testing and movement of the code through the pipeline. He also pointed out that these scripts need to be updated regularly once the system is changed and that the scripts also require the design of new general IT controls (GITC) around them to ensure that they are always up to date.

When asked about the importance of soft controls in Agile and DevOps, the interviewee pointed out that this creates a significant responsibility for the learning & development department to ensure that employees are trained properly and are continuously made aware of their responsibilities and the new processes. This could for example happen through regular (online) trainings and letting them sign an agreement when they start working at the company. He also stated that he believed the soft controls were the most important controls in this way of working because the other controls would only be effective if the soft controls were.

Concerning the authorization process of automated deployment of products he stated that it depends on the type of change that was conducted and that there is no one best way to manage this. While small changes like performance related updates could easily be automated and authorized by two developers, other changes that impact functionality and possibly have an influence on legal or security issues should be discussed with other knowledgeable actors. He also pointed out that departments such as legal and security should be involved in the design of the tests to ensure that only changes are deployed that comply with their standards. He furthermore stated that the "four eye principle" (separation of duties) could theoretically be replaced by one developer deploying a change alone and the pipeline taking the role of the second pair of eyes. Of course this would only be possible if the developer was not allowed to change the pipeline on his own and if the scripts were secured with a set of properly functioning GITC. He also thought that the most important control is that companies should not open their production for development activities because that would make all tests in the deployment pipeline redundant.

8.4.2 *DevOps risk governance*

The respondent liked the visualization of the risk governance house and agreed that culture was an important element that should represent the foundation of the house. Generally, he thought that the more a company depends on automated controls, the more important the human factor becomes. Similar to previous respondents, he had difficulties seeing the difference between the general risk management practices and control processes, although when explained to him, he thought that they were somewhat similar to the first and second line of defence in the three lines of defence risk management model. He generally thought that the controls were more operational than the risk management practices but pointed out that some practices like frameworks and project management influenced other components which was also noted by the first respondent. It therefore was ultimately decided to rearrange these pillars and to create three risk management pillars based on the three lines of defence and to position the pillar representing the DevOps components next to them.

8.4.3 *DevOps risk management*

The respondent agreed with the axes *"risk appetite"* and *"DevOps maturity"* on the risk management matrix and added that in order to achieve a high DevOps maturity, the fact whether the DevOps culture is already embedded in the minds of the teams plays a big role. He also agreed that the positioning on the matrix would not be the same throughout the company but depended on the system at hand and the team that was responsible for it. When confronted with the statement of the third respondent that companies in high risk environments should also consider completely automating their deployment, the respondent said that this was largely related to the risk appetite of a company. While some companies like WebTech1 and 2 would be quicker to choose automated deployment, teams working with systems related to airline security or treasury management systems would not want this. Furthermore, he pointed out that the identification of the impact of a change and the decision which scripts need to be run on the piece of code to be deployed was an action that needed to be done by a human being and could not be automated. He therefore thought that continuous deployment in these environment would require such an extremely high degree of preventive controls on the front-end that it would probably never be feasible for all companies. He also pointed out that companies who had to be SOx compliant were required to give out an in-control statement every year and therefore could not take any risks but had to be sure their operations remained in control during the DevOps transition. Looking at the proposed controls, he mentioned that asking a CAB for advice was still very traditional and that this should just happen if changes were deployed that were not in line with the beforehand agreed IT strategy.

8.5 case study participants

The results from the survey sent to the case study participants are summarized in the following. The survey was answered by six respondents.

8.5.1 *Risks in DevOps*

The respondents generally agreed that the overview of risks covered all important risks in DevOps. Only one respondent indicated that he had a neutral opinion concerning the overview and another respondent commented that quality risks were lacking. This was due to an error in the survey which did not mention quality risks as part of product risk. Since the respondent did not mention any other risks missing, it is assumed that this respondent also agrees with the overview of risks. Similarly to most interviewed experts, one respondent pointed out that project risks were not specifically associated with DevOps. He however mention that an advantage of agile projects is that teams aim at delivering a working piece of software every sprint, starting with the highest priority backlog items. This means that even if a project does not make the deadline, the teams will still have delivered some important functionality which is not necessarily the case in traditional projects. Another respondents thought that transitional risks were not very significant and that organizational risks (DevOps teams having to work with non-DevOps teams) was more of a bad practice than a risk. While this construction is undoubtedly a bad practice, it was decided to keep it as a risk in the overview since many companies in reality still struggle with this and therefore need to consider this construction as a source of risk.

8.5.2 *DevOps risk management and controls*

Opinions varied concerning the choice of the axes *"risk appetite"* and *"DevOps maturity"*. Employees from one company slightly disagreed with this choice whereas two other respondents agreed and one was neutral. Since the names of the axes were rated very positively by most interviewed experts, it was decided to keep these names. Two respondents remarked that the maturity of the DevOps teams was also dependent on the company as a whole and its culture and technical ability and is thus not entirely up to the teams. One other respondent thought that DevOps had different risks and a different view on risks than traditional methods and that the whole implementation of DevOps should be considered from this angle.

The vast majority of respondents (four of six) agreed that the proposed strategies and controls would address the most important DevOps risks whereby only one respondent was neutral and one disagreed. Similarly, four respondents thought that the proposed controls would not hinder the DevOps process unnecessarily while two respondents were neutral. Two respondents however remarked that the control *"all changes through CAB"* for the *"Agile teams responsible for Dev+Ops"* strategy was too restrictive. The control was therefore changed to *"most changes through CAB"* with the explanation that very low impact changes could be deployed by the teams directly. One respondent stated that the *"Experimental Learning"* strategy should contain a more controlled environment by using e.g. BlueGreen deployment and A/B testing. These controls were added to the strategy. The same respondent criticized the suggestion to let teams in this strategy report to management since this would undermine team responsibility and lead to responsibility-avoiding behaviour. As a response, a description was added to the strategy that this should only happen as long as the teams are not ready to work completely independently. Furthermore, reporting to management incidentally is expected to increase the sense of responsibility among team members since they are still free to make their own decisions but need to justify these towards the management. Another respondent from the same company did not agree that the *"Continuous deployment"* strategy was positioned as a higher risk appetite strategy since he thought that manually setting software into production brought about more risks. Since bringing the software into production only includes one click on the button and this strategy thus includes another preventive control, it was decided to keep the positions of strategies this way. The respondent added that an advantage of continuous deployment is that deployments can be planned at times when users are not disturbed by the updates whereas a manual deployment would likely take place during working hours.

Three of the respondents thought that the matrix was a useful tool for companies who did not know yet know how to organize their processes related to DevOps whereas one respondent was neutral and employees from one company disagreed. These were the two respondents who indicated that they did not agree with the names of the axes in the first place and subsequently did not agree very much with the proposed strategies.

Responses were mixed concerning the placement of risks on the matrix and to what extent the strategies sufficiently mitigated these risks. Two respondents thought that the strategies addressed the risks sufficiently while one was neutral and three disagreed. This was the only part of the risk management framework with which half of the respondents disagreed. While there was no explanation given by these respondents, it is assumed that respondents that did not agree with *"Continuous Deployment"* as a higher risk appetite strategy did also not agree that product risks were higher in companies that deployed more often.

8.5.3 *DevOps risk governance*

Multiple respondents indicated that they liked that culture was a fundamental part of the governance overview. One respondent stated that the house should also visualize how the DevOps processes connect with the risk management processes, e.g. through monitoring, logging and quality assurance in testing. While this was not possible to visualize due to aesthetic reasons, we added a description to the house that explains the interdependency of the two groups. Another respondent indicated that he needed some more explanation to fully understand the visualization. It can therefore be concluded that the overview of risk governance components should always be well explained to avoid misunderstandings.

8.6 summary and adjustments

The validation interviews have led to some minor changes such as renaming the *"risk impact"* axis to *"risk appetite"* and creating three pillars based on the three lines of defence in the risk governance house. Furthermore, some controls were slightly adjusted or added based on feedback from the case study respondents.

The respondents overall agreed with both the classification of risks as well as the risk management matrix and its respective strategies. Only case study participants from one company disagreed with the proposed strategies and one DevOps expert did not agree with the traditional controls. Overall, the matrix was evaluated positively on its risk mitigation ability and very positively on the agility requirement, meaning that the proposed controls sufficiently mitigated risks and would not hinder the DevOps process unnecessarily. Managers, auditors and IT Risk experts seemed to be more positive about the proposed approach than DevOps team members and one of the DevOps experts.

This expert and one survey respondent indicated that companies in high risk environments could also completely automate their deployment pipeline whereas the two risk management experts thought that this would theoretically be possible but require such an extensive amount of preventive, automated controls and trust into the pipeline that such a strategy would probably not be feasible for companies with a very low risk appetite. The other case study respondents and DevOps expert also indicated that they agreed with the positioning of strategies on the matrix. It was therefore decided to maintain the suggested continuous delivery approach for companies with a low risk appetite.

Many respondents noted that the project risk category was not entirely representative of DevOps organizations since projects are a traditional concept. Nevertheless, they acknowledged that many companies still use projects and this is therefore a source of risk that needs to be considered. One respondent pointed out that Agile projects also have some advantages compared to traditional projects, although a comparison of these two is not within the scope of this research. Due to the amount of companies that still use this term and in order to stress that risks in this category do not only apply to development activities, it was ultimately decided to keep the term "project" in the risk category overview. The category therefore describes any activity within the DevOps environment that is bound to a time planning, budget and involves multiple actors. One DevOps expert stated that the project risk category will vanish once companies completely embrace the DevOps mindset and stop working project-based.

Another important conclusion is that the risk categories identified are not mutually exclusive but represent different levels on which the risks can manifest themselves. Nevertheless, all respondents thought that the overview was a useful tool that could help companies identify relevant risks. Lastly, it was found that the DevOps risk governance overview should always be accompanied by a detailed explanation concerning the interdependencies of some components.

9 DISCUSSION

9.1 IMPLICATIONS

The objective of this research was to create a risk management framework that helps companies control their DevOps environment while remaining agile. In order to do so we have identified risks that are applicable to companies using DevOps as well as gathered and structured risk management practices that can be used for managing risks in DevOps. The encountered evidence from this study has wide reaching implications for companies, auditors as well as future research.

9.1.1 *Core values of DevOps and how to implement them*

The DevOps approaches presented in our framework are based on the four dimensions of Lwakatare et al. which are *collaboration*, *automation*, *measurement* and *monitoring*. As mentioned in Section 2.1, this thesis did consciously not choose one exact definition of DevOps and stick to it. The final artifact designed to help companies implement DevOps supports this decision since the "What is DevOps" discussions seems rather irrelevant considering the need for tailoring the concept to the companies' needs and situation. While some practitioners would only consider a completely automated CD strategy to fall under the definition of DevOps, others consider DevOps to only be the combination of *development* and *operations* into one team which is given in all four strategies. This diverse understanding of DevOps was also encountered during the case studies since companies had widely differing processes. Looking at the results of this research, the real question that companies should ask themselves is not how they can implement a fixed definition of DevOps they have in mind and simultaneously remain in control, but rather how they can tailor the DevOps philosophy to their needs in order to leverage their risk management processes through DevOps and experience the maximum amount of benefits without risking major losses or disasters.

Despite being averse to defining a precise definition however, this study has made clear what the core values of DevOps are and which elements are most important in a DevOps environment. It has become obvious that soft controls such as culture, communication and the feeling of responsibility are essential. These controls have a significant impact on the success and the risks of a DevOps endeavor and should not be underestimated, neither by managers nor by auditors. The cultural aspects encountered in this study can also be related back to the Agile manifesto which is the core of all agile methods and emphasizes *individuals and interactions, working software, customer collaboration* and *responding to change*.

Even if a company was technologically very advanced and had implemented the best automated controls available, a lack of responsibility or trust will still lead to the failure of the DevOps process. The direct consequence to these findings is that companies using DevOps will have to find a way to consolidate the DevOps culture in the organization, preferably before implementing any other DevOps processes. Employees need to be encouraged to come forward if they have made a mistake, knowing they will not be punished for it. Furthermore, companies need to ensure that teams communicate among each other and are aware of their responsibilities. Companies that aim for the maximum rate of agility should also encourage teams to take some risks while maintaining the continuous improvement mindset. On one hand this means that companies need to make sure the right processes and tools are in place, such as key performance indicators that focus on recovering fast from mistakes instead of failure rates as suggested by Farroha and Farroha [13], and the availability of tools that support communication between teams. It is also necessary to clearly assign and communicate responsibilities, both in terms of team responsibilities as well as by creating supporting structures outside the team e.g. for questions regarding security, compliance or legal issues. The more difficult part however includes changing the mindset of employees. Examples of mechanism for this are the "failure treats" encountered during the case study as well as regular trainings provided by the learning & development department and communication plans for scaling up problems.

9.1.2 *Risk management in DevOps and the role of automation*

The results further suggest that a crucial part of risk management concerns the process design of DevOps. Nevertheless, recurring risk evaluation dialogues are a very important practice to ensure that the designed process still addresses all relevant risks in the volatile DevOps environment. The use of an automated delivery pipeline increases the responsibility of the DevOps teams to consider the impact of a change before every deployment and to ensure that the automated tests cover all relevant parts of the code. Automated checks can again aid with this by halting the process if the test coverage is not high enough, however, they cannot replace human judgement. This stresses the importance of soft controls once more. Generally, automation of processes should be seen as a supporting mechanism to help achieve the DevOps values by organizations and not as a goal in itself. The case study of GeoTech (Section 6.6) shows that it is not necessary to completely automate the delivery process in order significantly improve incident resolution and change throughput times. The added value of DevOps rather seems to lie in the combination of development and operations into one, autonomous team instead of complete automation. Looking at the risk management framework, this indicates that companies should not stay in the low-maturity quadrants for too long since DevOps teams have to learn how to operate independently to reap the true benefits of DevOps. These strategies are therefore merely meant to support companies during their maturity growth.

This study also suggests that completely automating the deployment pipeline is not always feasible nor desirable for companies with a low risk appetite, although a few respondents working with DevOps did not agree with this statement. Some of these respondents pointed out that DevOps inherently mitigates risks by emphasizing rigorous testing and holding teams responsible for development and operations. Nevertheless, continuous deployment brings new challenges and advantages that need to be considered when choosing to implement it: Firstly, deploying small releases frequently, leads to earlier error detection and allows companies to perform faster rollbacks. Furthermore, employing automated checks leads to a basic quality assurance of the deployed product. On the other hand, frequent deployment also means that more items might be deployed that are not working as expected, especially if releases are committed that contain new functionality which is not (yet) covered by the automated tests. Therefore, IT Risk experts agreed that companies with a low risk appetite could benefit from adding another manual approval before deployment (as suggested in Section 7.3.2) while still making use of automated testing or could alternatively aim at creating a hybrid environment as proposed by Yasar [56].

One of the most difficult parts of creating an efficient DevOps environment seems to be the integration of processes from DevOps teams with non-DevOps departments. According to one interviewed expert, it is most desirable that an organization completely moves towards a DevOps way of working instead of trying to keep traditional structures. However, this might not be feasible for all organizations. Alternatively, organizations could consider separating DevOps teams from the rest of the organization and treat them like separate entities or internal start-ups, therefore limiting their contact with non-DevOps departments.

Despite the challenges encountered, all case study respondents were positive about using DevOps which suggests that the DevOps approach yields some significant advantages over traditional development methods such as faster delivery speed, less incidents and improved customer service. This was also clearly demonstrated in the case study of GeoTech in Section 6.6. The risk management framework has therefore been designed in such a way that it supports and accompanies the companies throughout their maturity growth since higher DevOps maturity will let them reap more benefits. Contrary to some expectations created in literature, most case study participants did not perceive compliance requirements as hindering the DevOps process particularly. However, the described situations are not entirely comparable since Laukkarinen et al. [22] performed an analysis of requirements for medical devices. Compliance therefore only seems to be an obstacle to DevOps in very specific sectors but not generally to the broad range of companies which we studied. Most importantly, this research has shown that it is entirely possible to remain in control of DevOps and CD practices and to demonstrate this control towards auditing parties if DevOps is implemented correctly. Generally, this becomes easier as more parts of the pipeline are automated while hybrid environments are more difficult to manage. Some tools can even improve compliance and control significantly compared to traditional methods through the use of automated logging tools as well as IaC and containers that increase traceability and ensure correct configuration of the development, testing and production environments at all times.

9.1.3 *The future of IT audit*

As predicted by literature and interviewed experts, DevOps will continuously gain importance in IT audits due to its increasing application on critical enterprise systems. This makes it imperative for companies using DevOps to be capable of demonstrating the control of their processes. However, this research also shows that the profession of IT auditors will change significantly in the future. Besides having to audit the effectiveness of soft controls within an organization, auditors will also have to become more technologically skilled. Instead of auditing procedures and samples, IT audit will have to focus on automation tools and scripts that execute these procedures. This requires that auditors can e.g. understand deployment scripts and assess their effectiveness, evaluate automated tests and review log files. Furthermore, they will have to assess the GITC around the deployment pipeline in terms of access and change management.

9.2 VALIDITY AND RELIABILITY OF RESEARCH

Reliability refers to consistency of measures, while *validity* is characterized as whether an indicator that is designed to measure a certain concept really does measure the intended concept [3]. According to Yin [57], case study research has to maximize four subtypes of reliability and validity to ensure the quality of the design: *construct validity*, *internal validity*, *external validity*, and *reliability*. These will be discussed in the following.

9.2.1 *Construct validity*

Construct validity implies that the researcher needs to establish correct operational measures for the concepts being studied [57]. Construct validity is sometimes viewed as problematic in case study research since some subjective judgement is used to collect the data.

The initial measures collected during the case studies in this research concerned process design and practices such as access management, change management and the configuration of the software delivery pipelines. These concepts were therefore "hard facts" that could be objectively measured and had shared meaning by all participants involved. Concepts that were more difficult to measure include softer aspects related to the success of risk management or DevOps, responsibility and trust since these depend on personal judgement of the interviewee and researcher.

Riege [38] suggests to ensure construct validity in case study research by *using multiple sources of evidence*, *establishing a chain of evidence* and to have *key informants review the draft case study report*. One limitation to the first criterion is that various case studies depended on one single interview with one employee of the company. On the other hand, we have studied a broad range of organizations and used multiple types of sources (interviews, documents, observations and informal conversations) in other cases, which compensates for the thin evidence within some companies. The second criterion was fully fulfilled since a chain of evidence should be established by using verbatim interview transcripts and notes of observations which supply sufficient citations and cross checks of sources of evidence. Lastly, all all informants have read their respective interview transcript, however, the review of the results through case study participants was limited to six people. Overall, it can therefore be concluded that the three criteria suggested by Riege have been sufficiently fulfilled although the limited number of evidence within some case studies partly limits construct validity.

9.2.2 *Internal validity*

Internal validity is of concern to causal or explanatory case studies in which the researcher tries to determine whether event x led to event y [57]. The research at hand draws many causal conclusions such as the influence of certain controls on successful risk management. Internal validity of these conclusions is therefore of great importance to this research. According to Riege [38], the corresponding design test to internal validity is credibility. Yin [57] suggests the following case study tactics for increasing internal validity: *pattern-matching, explanation-building* and *time-series analyses*. While this research does not draw on time-series analyses, the other two criteria were fulfilled. Pattern matching was conducted by studying multiple companies and observing which features and processes companies with mature DevOps processes shared. The factors that were identified as critical for managing risks in DevOps were then provided with detailed explanations. Furthermore, the collected data was analyzed and relationships were established using open and axial coding techniques. This way we have also established within-case as well as cross-case analyses of the available data as suggested by Riege [38]. It can therefore be concluded that there are little doubts concerning the internal validity of this research.

9.2.3 *External validity*

External validity is concerned with establishing a domain to which a study's findings can be generalized [57]. The study at hand has tried to differentiate the conclusions based on the companies risk appetite and DevOps maturity and therefore aims to represent a range of companies and establish wide generalizability. According to Yin [57], using a replication logic and multiple case studies as done in this research can help increase external validity and thus ensure that the findings are generalizable beyond the case study at hand. We have also compared the collected data to evidence from extant literature as suggested by Riege [38].

Nevertheless, it is never possible to be completely certain of external validity, even though we have applied rigorous research methods and replicated some findings across multiple companies. The external validity of this study is limited by the number of case studies which is extremely small compared to all companies using DevOps. Furthermore, all case companies were located in the Netherlands which could also lead to cultural bias. It is therefore not possible to guarantee full external validity, although the research methods and findings suggest a rather high degree of generalizability.

9.2.4 *Reliability*

Reliability is characterized as demonstrating that the operations of a study can be repeated with the same results[57]. In order to do so, case study researchers should use a case study protocol and develop a case study data base [57]. Among others, Riege [38] suggests to give full account of theories and ideas and assure congruence between the research issues and features of the study design.

The conducted research phases have been recorded as detailed as possible in this thesis. Due to the anonymity of case study respondents it is not possible to exactly describe the nature of each company which somewhat limits replicability. Although almost every company implemented DevOps differently, it is expected that a second execution of this research will encounter similar risk management practices as we have in this research. This is especially plausible for those practices that were encountered in multiple companies. It is also expected that the same soft aspects will be identified as important as in this research.

9.2.5 *Design research quality*

The above performed analysis was based on validity and reliability criteria for case study research. In order to demonstrate the validity and reliability of the design research part of this study, we use the seven guidelines for design-science research by Hevner et al. [18]. When analyzing the research using these guidelines, it becomes evident that all principles have been fulfilled. The weakest part of the research is the validation (Guideline 3) since this was only done with expert opinions and a limited number of case study respondents. A truly naturalistic validation of the artifact, for example through a case study, is therefore lacking and could be conducted in future research. However, all other guidelines have been rigorously followed which suggests an overall reliable and valid design research.

Table 9.1: Guidelines for design-science research according to Hevner et al. [18] applied to this research

GUIDELINE		DESCRIPTION	APPLICATION
1	Design as Artifact	Design-science research must produce a viable artifact in the form of a construct, a model, a method or an instantiation.	We provide a DevOps risk management matrix including four strategies for managing risks in DevOps.
2	Problem Relevance	The objective of design-science research is to develop technology-based solutions to important and relevant business problems.	Risk management in DevOps has been identified as a relevant business problem. Our solution strategies are mostly technology based.

GUIDELINE		DESCRIPTION	APPLICATION
3	Design Evaluation	The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods.	Our framework was validated with expert opinions and the case study respondents. An empirical validation is still lacking.
4	Research Contributions	Effective design science research must provide clear and verifiable contributions in the areas of the design artifact, design foundations and/or design methodologies.	This research is one of the first studies to explore the topic of risk management in DevOps. It is also one of a limited numbers of empirical studies in the domain of DevOps.
5	Research Rigor	Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artifact.	We have conducted a rigorous multivocal literature study according to the procedure by Kitchenham [21] and empirical case studies applying a structured coding process [3].
6	Design as a Search Process	The search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem environment	The design and validation process took iterative approaches in search of the optimum artifact.
7	Communication of Research	Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences	The research has been presented to a group of IT Audit & Risk Management professionals as well as to a group of IT Managers and CIOs. It will also be defended in front of university researchers in the field of Business Information Technology.

9.2.6 *Summary*

While no major threats to internal validity, external validity and reliability have been identified, case study research is often naturally limited in construct validity due to subjective measurements. In our research, many of the measured objects were processes and practices which can be objectively measured. However, aspects related to organizational culture were more difficult to identify and are therefore limited in construct validity. Although we have taken measures to limit subjectiveness by collecting data from multiple sources, some cases depend on only one interview with a single employee and are therefore still limited in validity. This is somewhat compensated for by the fact that we have studied a total of nine companies.

A second limitation is that this study does not provide a completely naturalistic evaluation. In order to verify whether the framework really brings the expected results, a case study should be conducted by using the framework while implementing DevOps in a real-world company.

9.3 CONTRIBUTIONS TO RESEARCH AND PRACTICE

9.3.1 *Contributions to research*

Literature related to risk management in DevOps is still limited and strongly lacks empirical research. The lack of research on this topic has even been lamented by interviewees during this study. Nevertheless, research about DevOps in general has increased significantly over the past two years and is expected to grow further as the DevOps concept continues to gain importance in the business world. This study is also one of the first to provide empirical evidence on managing risks in DevOps. We have set an example for subsequent research by studying a total of nine companies and providing an overview of their practices. Due to the empirical component, this study also provides less idealized descriptions of DevOps processes and more real-world insights. Most other papers we encountered only described very advanced, automated processes which were only encountered in few companies during this study and did not address the real problems that most companies were struggling with. Furthermore, only a small amount of papers have been identified that specifically dealt with addressing risks in a DevOps context of which the thesis at hand is the most elaborate research. We therefore believe that this research contributes significantly to the DevOps research community.

Besides researching risk management in DevOps, we have also described the case of a company which implemented DevOps and provided quantitative data about the observed benefits. This was something that was still missing in academic literature according to Erich et al. [12].

9.3.2 *Contributions to practice*

On the practical side, this thesis provides a framework based on proven empirical practices which can guide DevOps practitioners in designing appropriate DevOps processes for their companies and addressing relevant risks. The use of such a framework has been acknowledged by case study participants and industry experts which sparks the hope that the framework at hand will be of real use to them.

On a general note, this thesis might contribute to taking the "buzz" from the DevOps hype and help DevOps advance towards a useful concept that can actually add business value. For this to happen, practitioners need to realize that only focusing on the implementation of trending topics like continuous deployment pipelines is not always the best solution for their company as demonstrated in this thesis.

9.4 RELATED AND FUTURE WORK

9.4.1 *Related work*

During the course of this research, more papers in the domain of DevOps have been published which have not been included into the literature review. Our claims towards the academic contributions of this paper are therefore limited to the time of the literature data collection which was conducted in June 2018. An exploratory literature review has shown that as of February 2019, notable academic contributions about compliance and DevOps as well as DevSecOps include:

- V. Mohan, L. ben Othmane and A. Kres "BP: Security concerns and best practices for automation of software deployment processes -An industrial case study", *2018 IEEE Secure Development Conference* [31]
- A. Subramanian, P. Krishnamachariar, M. Gupta and R. Sharman, "Auditing an agile development operations ecosystem", *International Journal of Risk and Contingency Management* [48]

  -this paper was not fully accessible and was selected based on the abstract and introduction
- T.C.M. Fernandes, I. Costa, N. Salvetti, F.L.F. de Magalhães and A.A. Fernandes, "Influence of DevOps practices in IT management processes according to the COBIT5 model", *NAVUS – Revista de Gestão e Tecnologia* [14]

  -this paper is in Portuguese and was only selected based on the English abstract
- L. Williams, "Continuously integrating security", *Proceedings - 2018 ACM/IEEE 1st International Workshop on Security Awareness from Design to Deployment, SEAD 2018* [55]

During this exploratory literature review, no papers covering the exact same problem statement as this research have been encountered which shows that the relevance of this study remains.

9.4.2 *Future work*

This thesis has only studied the implementation and execution of DevOps from a risk management perspective. In order to guarantee successful DevOps processes, more research is needed to identify other factors that ensure implementation success. Likewise, this thesis only provides a limited overview of benefits associated with DevOps (called "success indicators" in this thesis) whereas a more complete assessment still needs to be established.

An ongoing challenge for practitioners is the application of the so-called "soft controls" within an enterprise such as company culture, employee responsibility and trust. Future research should focus on how to implement these control as well as how to demonstrate them so they could potentially be considered in DevOps audit and assurance engagements in the future. The thesis at hand has already made a start by naming some practices that could indicate the presence of these controls such as security awareness trainings, non-disclosure agreements or team responsibility trainings. Furthermore, it was stated during the literature review that existing compliance requirements like frameworks and laws are mostly directed at traditional development processes and therefore do not fit naturally with DevOps. In order to provide assurance on the increasing amount of agile and DevOps processes, these frameworks will have to be adjusted in the near future. This thesis has already demonstrated how DevOps processes can be kept under control and suggested some fitting controls for this.

Lastly, this study was not concerned with comparing the risks from a traditional waterfall method and the DevOps approach. Some respondents thought that DevOps inherently lowered risks and that risks in DevOps were therefore fewer and different than risks in waterfall. Other respondents pointed out that the overview of risks given in Section 7.1.1 showed that risks in DevOps are not very different from those encountered in a traditional waterfall process and that only the controls to mitigate these risks are different. Since we did not study the waterfall method in this study, we can make no conclusions regarding a comparison between the two approaches. However, a comparison is also not expected to be of importance to the target companies of this study since they have already decided to transition to DevOps and are not concerned with the waterfall method anymore. For companies still contemplating whether to implement DevOps or not, a thorough comparison of risks in waterfall and DevOps and a conclusion which approach bears the most overall amount of risks however could be of value.

10 CONCLUSION

10.1 RESEARCH QUESTIONS

This study aimed at investigating possibilities for companies to manage risks and increase internal control while using DevOps without trading off too much of the agility and benefits that DevOps offers. Secondly, it intended to research how these companies can demonstrate their internal control towards IT auditing parties. It has followed the main research question:

What is a suitable framework that allows companies to mitigate risks and exercise control over their DevOps environment while remaining agile?

In order to answer this question, a multivocal literature review and nine case studies have been performed. The results of this study have been evaluated with four experts and six case study respondents. The main research question was divided into three sub-questions which are answered in the following.

1. What types of risks are companies using DevOps exposed to?

The empirical research has shown that risks associated with DevOps can be grouped into five categories which are *transitional, organizational, project, team* and *product* risks. As experts pointed out during the validation phase, not all risks in these categories are purely associated with DevOps since they are also important in traditional forms of software development. Furthermore, the risk categories are naturally overlapping since one risk can be associated with multiple categories and one risk can lead to another. Project risks and transitional risks are expected to only apply to companies who are still in particular stages of the DevOps transition while the other risks remain relevant to all companies using DevOps. The classification was judged to be a helpful tool for companies when analyzing risks in DevOps.

2. Which practices exist that can be incorporated into a DevOps process to demonstrate control and ensure the creation of valid audit trails?

This thesis has given an extensive overview of practices that can be applied in DevOps to address the risks identified in research sub-question 1. These practices concerned both automated controls that are integrated into the DevOps process as well as more traditional controls that can be used in combination with DevOps. The hard controls were categorized into *access management, change control, security, compliance, monitoring, logging* and *others*. Besides these hard controls, important soft controls such as culture, communication and team responsibility and more general risk management aspects and frameworks were identified. A complete overview of all practices can be found in Appendix D.

3. Which strategy should companies drive in order to identify risks and implement suitable controls?

Based on the findings related to research sub-questions 1 and 2, four strategies were designed which teams can use based on their DevOps maturity and risk appetite. While only one of these strategies would be considered as "fully DevOps" (referring to complete automation of processes and maximum speed) by some practitioners, all strategies allow users to reap the maximum amount of benefits from DevOps while remaining in control of their operations. We also show that complete automation of processes is not necessary in order to significantly improve change throughput and incident resolution times since the main added value seems to lie in the combination of developmental and operational tasks.

10.2 KEY CONTRIBUTIONS AND FINDINGS

During the literature review, a lack of empirical research on risk management and DevOps was encountered. The research at hand therefore helps both practitioners as well as researchers to understand the interdependency of risk management and DevOps better. Our findings generally align with the scarce resources that were available. The main contributions to academia and practice are threefold:

- *scientific*: This study is one of few and the most elaborate empirical study we have found so far regarding risk management in DevOps.
- *practical*: We provide an overview of important DevOps risk governance components and a contingency-based framework for companies on how to integrate risk management practices into their DevOps processes.
- *practical*: This study provides insights for auditors that want to provide assurance on internal control on how to audit DevOps processes.

The most important findings of this research are outlined in the following section by means of key statements.

1. DevOps does not have to be a trade-off between risk and agility

This research has demonstrated that applying DevOps practices does not have to lead to less control over processes. Case studies have shown that full or partial automation of the deployment pipeline and infrastructure can help increase control by enforcing automated tests, automatically monitoring and logging actions and using version control on code and infrastructure configurations. However, companies that do not want to automate their processes completely can also remain in control while using DevOps by integrating more traditional, manual practices into the delivery process.

2. DevOps culture is the basis for successful risk management

We have found that a culture of responsibility, communication and trust is essential for managing risks in DevOps. While this is probably the most difficult part to implement during the transition, DevOps heavily relies on these aspects since no elaborate approval processes are conducted and teams can often deploy releases independently. Among others, this implies that teams need to be responsible enough to come forward in case of questions or concerns. Nevertheless, teams also need to receive enough autonomy from management to solve incidents quickly and independently.

3. DevOps requires clear roles and responsibilities

DevOps requires clearly assigned roles and responsibilities. This refers to the responsibilities of the teams as well as to contact people outside of the teams whom they can approach in case of questions or concerns. Second line of defence functions are of great importance in DevOps since these need to ensure that the teams are informed sufficiently e.g. about security or compliance issues and need to provide the teams with the right tools. They should also audit the teams regularly without undermining their autonomy. Different to traditional processes, these roles and responsibilities however should be embedded into the minds of the employees instead of just being defined in procedure files.

4. There is no one best way to manage risks in DevOps

Two factors were identified that influence how companies organize their software delivery processes:

- *Risk appetite*: The extent to which a company accepts that some releases are deployed that are not completely accurate and prefers to perform fast rollbacks instead of not making any mistakes.
- *DevOps maturity*: How independently the teams are able to operate as well as to what extent they are ready to automate parts of their delivery process.

While companies with a lower risk appetite used more preventive controls, companies with a higher risk appetite were more prone to use continuous deployment techniques. Furthermore, many companies made a distinction between low impact changes that can be deployed independently by the teams and higher impact changes that have to be approved by a Change Advisory Board first. The proposed risk management framework takes these influencing factors into account by suggesting extra manual controls for teams that are not yet using elaborate automation and proposing more preventive controls for companies that do not want to rely purely on a continuous deployment process.

5. Continuous deployment is not for everyone

While some companies stated to aim for complete automation of their deployment process, others did not think that deploying multiple times per day was necessary. Continuous deployment brings upon various advantages and challenges and it is ultimately up to the company to decide whether it wants to accept these. Nevertheless, automation should rather be seen as a supporting mechanism to implementing the DevOps philosophy by aiding faster and more frequent deployment and ensuring basic quality assurance through automated testing.

6. Opinions about DevOps differ widely

All interviewees were generally positive about using DevOps, although people had widely differing opinions about what DevOps was and how it should be dealt with. It therefore seems unattainable to develop a framework that everyone involved with DevOps completely agrees with. One expert and some case study respondents regarded DevOps as inherently mitigating risks and thus being a lower-risk approach than waterfall methods. Other respondents thought that DevOps was a method with great potential that however needed thorough risk management practices to remain in control. While some validation respondents working with DevOps thought that companies with a lower risk appetite should use continuous deployment since this would bear less risk, other respondents among which IT Risk experts thought that continuous delivery was a feasible alternative due to the extra manual approval. Ultimately, the choice which risk mitigation strategy to use is of course left to the company.

BIBLIOGRAPHY

[1] Addison-Hewitt Associates, "A guide to the Sarbanes-Oxley Act," 2006, "[Online]. Available: http://www.soxlaw.com. [Accessed: 12-Jan-2019]".

[2] R. Bierwolf, P. Frijns, and P. van Kemenade, "Project management in a dynamic environment: Balancing stakeholders," in *2017 IEEE European Technology and Engineering Management Summit (E-TEMS)*, 2017, pp. 1–6, ISBN: 978-1-5386-3721-0. DOI: 10.1109/E-TEMS.2017.8244226.

[3] A. Bryman, *Social research methods*, 4th. Oxford University Press, 2012, ISBN: 978–0–19–958805–3.

[4] Committee of Sponsoring Organizations of the Treadway Commission, *Enterprise risk management - Integrated framework*, 2004.

[5] H. Davies and M. Zhivitskaya, "Three lines of defence: A robust organising framework, or just lines in the sand?" *Global Policy*, vol. 9, no. Supplement 1, pp. 34–42, 2018, ISSN: 17585899. DOI: 10.1111/1758-5899.12568.

[6] J. DeLuccia IV, J. Gallimore, G. Kim, and B. Miller, "DevOps audit defense toolkit," IT Revolution, Tech. Rep., 2015.

[7] P. Debois, "DevOps: A software revolution in the making?" *Cutter IT Journal*, vol. 24, no. 8, pp. 3–5, 2011, ISSN: 15227383.

[8] DevOps Agile Skills Association (DASA), "12 critical skills and knowledge areas required for DevOps," "[Online]. Available: https://www.devopsagileskills.org/dasa-competence-model/. [Accessed: 20-Feb-2019]".

[9] DevOps Enterprise Forum, *An unlikely union: DevOps and audit - Information security and compliance practices*. IT Revolution, 2015.

[10] DevOps.com, "The state of DevOps adoption and trends in 2017," 2017, "[Online]. Available: https://devops.com/state-devops-adoption-trends-2017. [Accessed: 10-Jan-2019]".

[11] O. Diaz and M. Muñoz, "Fortaleciendo un enfoque DevOps con seguridad y gestión de riesgos: Una experiencia de su implementación en un centro de datos en una organización mexicana [Reinforcing DevOps approach with security and risk management: An experience of implementing it in a data center of a mexican organization]," in *2017 6th International Conference on Software Process Improvement (CIMPS)*, 2017, pp. 1–7, ISBN: 978-1-5386-3230-7. DOI: 10.1109/CIMPS.2017.8169957.

[12] F. M. Erich, C. Amrit, and M. Daneva, "A qualitative study of DevOps usage in practice," *Journal of Software: Evolution and Process*, vol. 29, no. 6, 2017, ISSN: 20477481. DOI: 10.1002/smr.1885.

[13] B. S. Farroha and D. L. Farroha, "A framework for managing mission needs, compliance, and trust in the DevOps environment," in *Proceedings - IEEE Military Communications Conference MILCOM*, 2014, pp. 288–293, ISBN: 9781479967704. DOI: 10.1109/MILCOM.2014.54.

[14] T. C. M. Fernandes, I. Costa, N. Salvetti, F. L. F. de Magalhães, and A. A. Fernandes, "Influência das práticas do DevOps nos processos de gestão de TI conforme o modelo COBIT 5 [Influence of DevOps practices in IT management processes according to the COBIT 5 model]," *NAVUS – Revista de Gestão e Tecnologia*, pp. 20–31, 2018, ISSN: 22374558. DOI: 10.22279/navus.2018.v8n1.p20-31.584.

[15] S. D. Gantz, *The basics of IT audit: Purposes, processes, and practical information*. 2013, pp. 1–244, ISBN: 9780124171596. DOI: 10.1016/C2013-0-06954-X.

[16] V. Garousi, M. Felderer, and M. V. Mäntylä, "The need for multivocal literature reviews in software engineering," in *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering - EASE '16*, 2016, pp. 1–6, ISBN: 9781450336918. DOI: 10.1145/2915970.2916008.

[17] V. Garousi, M. Felderer, and M. V. Mäntylä, *Guidelines for conducting multivocal literature reviews in software engineering*, 2017.

[18] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design Science in IS Research," *MIS Quarterly*, vol. 28, no. 1, pp. 75–105, 2004.

[19] IT Governance Institute, *IT control objectives for Sarbanes-Oxley [exposure draft]*, 2nd ed. 2006.

[20] R. Jabbari, N. bin Ali, K. Petersen, and B. Tanveer, "What is DevOps? A systematic mapping study on definitions and practices," *Proceedings of the Scientific Workshop of XP2016*, pp. 1–11, 2016, ISSN: 07421222. DOI: 10.1145/2962695.2962707.

[21] B. Kitchenham, "Procedures for performing systematic reviews," Software Engineering Group, Keele University & Empirical Software Engineering, National ICT Australia Ltd., Tech. Rep., 2004.

[22] T. Laukkarinen, K. Kuusinen, and T. Mikkonen, "DevOps in regulated software development: Case medical devices," in *2017 IEEE/ACM 39th International Conference on Software Engineering: New Ideas and Emerging Technologies Results Track (ICSE-NIER)*, IEEE, May 2017, pp. 15–18, ISBN: 978-1-5386-2675-7. DOI: 10.1109/ICSE-NIER.2017.20.

[23] T. Laukkarinen, K. Kuusinen, and T. Mikkonen, "Regulated software meets DevOps," *Information and Software Technology*, vol. 97, pp. 176–178, 2018, ISSN: 09505849. DOI: 10.1016/j.infsof.2018.01.011.

[24] A. Lichtenberger, "Fünf kritische erfolgsfaktoren für eine erfolgreiche DevOps transformation [Five critical DevOps success factors]," *HMD Praxis der Wirtschaftsinformatik*, vol. 54, no. 2, pp. 244–250, 2017, ISSN: 1436-3011. DOI: 10.1365/s40702-017-0293-6.

[25] L. E. Lwakatare, P. Kuvaja, and M. Oivo, "Dimensions of DevOps," *Lecture Notes in Business Information Processing*, vol. 212, pp. 212–217, 2015, ISSN: 18651348. DOI: 10.1007/978-3-319-18612-2_19.

[26] L. E. Lwakatare, P. Kuvaja, and M. Oivo, "An exploratory study of DevOps: Extending the dimensions of DevOps with practices," in *ICSEA 2016 : The Eleventh International Conference on Software Engineering Advances*, Rome, Italy, 2016, pp. 91–99, ISBN: 9781612084985.

[27] L. E. Lwakatare, P. Kuvaja, and M. Oivo, "Relationship of DevOps to Agile, Lean and Continuous Deployment: A multivocal literature review study," in *Lecture Notes in Computer Science*, vol. 10027 LNCS, 2016, pp. 399–415, ISBN: 9783319490939. DOI: 10.1007/978-3-319-49094-6_27.

[28] J. R. Michener and A. T. Clager, *Mitigating an oxymoron: compliance in a DevOps environment*, 2016. DOI: 10.1109/COMPSAC.2016.155.

[29] M. B. Miles and A. M. Huberman, *Qualitative data analysis: An expanded sourcebook*. SAGE, 1994, ISBN: 0803955405.

[30] V. Mohan and L. ben Othmane, "SecDevOps: Is it a marketing buzzword? Mapping research on security in DevOps," in *Proceedings - 2016 11th International Conference on Availability, Reliability and Security, ARES 2016*, 2016, pp. 542–547, ISBN: 9781509009909. DOI: 10.1109/ARES.2016.92.

[31] V. Mohan, L. ben Othmane, and A. Kres, "BP: Security concerns and best practices for automation of software deployment processes -An industrial case study," *2018 IEEE Secure Development Conference*, pp. 21–28, Sep. 2018. DOI: 10.1109/SecDev.2018.00011.

[32] M. Muñoz and O. Díaz, "DevOps: Foundations and its utilization in data center," in *Engineering and Management of Data Centers*, Springer, Cham, 2017, pp. 205–225. DOI: 10.1007/978-3-319-65082-1_10.

[33] P. A. Nielsen, T. J. Winkler, and J. Nørbjerg, "Closing the IT development-operations gap: The DevOps knowledge sharing framework," in *CEUR Workshop Proceedings*, B. Johansson, Ed., vol. 1898, CEUR, 2017, ISBN: 16130073.

[34] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *Journal of Management Information Systems*, vol. 24, no. 3, pp. 45–77, 2008, ISSN: 0742-1222. DOI: 10.2753/MIS0742-1222240302.

[35] B. Phifer, "Next-generation process integration: CMMI and ITIL do DevOps," *Cutter IT Journal*, vol. 24, no. 8, pp. 28–33, 2011, ISSN: 15227383.

[36] S. Pittet, "Continuous Integration vs. Continuous Delivery vs. Continuous Deployment," "[Online]. Available: https://www.atlassian.com/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment. [Accessed: 04-Feb-2019]".

[37] O. Plant, "Risk management in devops: A state of the art literature review - Research topics business information technology," *University of Twente*, 2018.

[38] A. M. Riege, "Validity and reliability tests in case study research: A literature review with "hands-on" applications for each research phase," *Qualitative Market Research: An International Journal*, vol. 6, no. 2, pp. 75–86, 2003, ISSN: 13522752. DOI: 10.1108/13522750310470055.

[39] RightScale, "Cloud computing trends: 2018 state of the cloud survey," 2018, "[Online]. Available: https://www.rightscale.com/blog/cloud-industry-insights/cloud-computing-trends-2018-state-cloud-survey. [Accessed: 10-Jan-2019]".

[40] A. Robinson, "Continuous security: implementing the critical controls in a DevOps environment," *SANS Institute InfoSec Reading Room*, 2015.

[41] A. Robinson, "A checklist for audit of Docker containers," *SANS Institute InfoSec Reading Room*, 2016.

[42] M. Rubino, F. Vitolla, and A. Garzoni, "The impact of an IT governance framework on the internal control environment," *Records Management Journal*, vol. 27, no. 1, pp. 19–41, 2017, ISSN: 09565698. DOI: 10.1108/RMJ-03-2016-0007.

[43] D. Shackleford, "A DevSecOps playbook," *SANS Institute*, 2016.

[44] J. Smeds, K. Nybom, and I. Porres, "DevOps: A definition and perceived adoption impediments," *Lecture Notes in Business Information Processing*, vol. 212, pp. 166–177, 2015, ISSN: 18651348. DOI: 10.1007/978-3-319-18612-2_14.

[45] K. Soin and P. Collier, "Risk and risk management in management accounting and control," *Management Accounting Research*, vol. 24, no. 2, pp. 82–87, 2013, ISSN: 10445005. DOI: 10.1016/j.mar.2013.04.003.

[46] D. Ståhl, T. Mårtensson, J. Bosch, 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, ISBN: 978-1-5386-2141-7. DOI: 10.1109/SEAA.2017.8114695.

[47] E. Střihavková, "Analysis of the status and quality of internal audit in selected organizations," *IOP Conference Series: Materials Science and Engineering*, vol. 393, no. 1, 2018, ISSN: 1757899X. DOI: 10.1088/1757-899X/393/1/012115.

[48] A. Subramanian, P. K. Krishnamachariar, M. Gupta, and R. Sharman, "Auditing an Agile development operations ecosystem," *International Journal of Risk and Contingency Management*, vol. 7, no. 4, pp. 90–110, 2018, ISSN: 2160-9624. DOI: 10.4018/IJRCM.2018100105.

[49] V. W. Tai, Y. H. Lai, and T. H. Yang, "The role of the board and the audit committee in corporate risk management," *North American Journal of Economics and Finance*, 2018, ISSN: 10629408. DOI: 10.1016/j.najef.2018.11.008.

[50] J. Venable, J. Pries-Heje, and R. Baskerville, "FEDS: a framework for evaluation in design science research," *European Journal of Information Systems*, vol. 25, no. 1, pp. 77–89, Jan. 2016, ISSN: 1476-9344. DOI: 10.1057/ejis.2014.36.

[51] P. Verschuren and H. Doorewaard, *Het ontwerpen van een onderzoek [Designing a research project]*, 5th ed. Amsterdam: Boom Lemma Uitgevers, 2015.

[52] M. Virmani, "Understanding DevOps & bridging the gap from Continuous Integration to Continuous Delivery," in *5th International Conference on Innovative Computing Technology, INTECH 2015*, 2015, pp. 78–82, ISBN: 9781467375504. DOI: 10.1109/INTECH.2015.7173368.

[53] A. Wiedemann, "IT governance mechanisms for DevOps teams: How incumbent companies achieve competitive advantages," *51st Hawaii International Conference on System Sciences*, vol. 9, pp. 4931–4940, 2018.

[54] R. J. Wieringa, *Design science methodology for information systems and software engineering*, 1st. Springer-Verlag Berlin Heidelberg, 2014, ISBN: 9783662438398. DOI: 10.1007/978-3-662-43839-8.

[55] L. Williams, "Continuously integrating security," *Proceedings - 2018 ACM/IEEE 1st International Workshop on Security Awareness from Design to Deployment, SEAD 2018*, pp. 1–2, 2018. DOI: 10.1145/3194707.3194717.

[56] H. Yasar, "Implementing secure DevOps assessment for highly regulated environments," in *Proceedings of the 12th International Conference on Availability, Reliability and Security - ARES '17*, Reggio Calabria, Italy: ACM, 2017, pp. 1–3, ISBN: 9781450352574. DOI: 10.1145/3098954.3105819.

[57] R. K. Yin, "Case study research: Design and methods," *Applied Social Research Methods*, vol. 5, 1994. DOI: 10.1177/017084068600700114.

APPENDICES

A STRUCTURED LITERATURE REVIEW: SEARCH PROTOCOL

A.1 INCLUSION AND EXCLUSION CRITERIA

INCLUSION	EXCLUSION
Papers about governance mechanisms within DevOps	Papers only about DevOps tooling
Papers about managing risks in DevOps	Papers only about DevOps infrastructure
Papers about achieving compliance in DevOps	
Paper about process management in DevOps	

A.2 SEARCH RESULTS

Search in academic databases

Databases: Scopus, Web of Science, IEEE XPlore, ACM Digital library (Guide to computing literature), AIS Electronic library

Search key: *"devops" AND ("risk management" OR compliance OR governance OR "IT controls")*

Search conducted on 21.06.2018

SEARCH PROCESS	AMOUNT OF PAPERS
Unique results	128
After filtering on title	33
After filtering on abstract	18
After filtering on full text	9
Adding forward & backward references	11

Grey literature search

Database: SANS InfoSec Reading Room (search via Google)

Search key: *site:https://www.sans.org/reading-room filetype:pdf "devops" AND ("risk management" OR compliance OR governance OR "IT controls")*

SEARCH PROCESS	AMOUNT OF PAPERS
Unique results	137
After filtering on title	4
After filtering on abstract	2
After filtering on full text	2
Adding forward & backward references	5

B STRUCTURED LITERATURE REVIEW: RESULTS

B.1 SELECTED PAPERS

The papers selected for the literature review are summarized in Table B.1. Next to a short description, the type of research is mentioned and whether the conclusions have been validated. The table also indicates the type of the paper. Nine papers are conference papers (C) and five papers are white papers (W). Only one journal paper (J) and one book chapter (B) were included.

Table B.1: Papers selected for literature review

PAPER	TYPE	DESCRIPTION
[2]	C	Analyze risk management capability of the DevOps approach. Propose a framework for comparing management and control measures by dividing them into four areas of attention. Conclusions based on literature study.
[6]	W	Guideline for auditors. Techniques on mitigating risks and auditing a DevOps process with a fictitious example. Authors claim to summarize techniques encountered while studying "a number of organizations", no further information.
[9]	W	Addresses three major concerns about DevOps: change control, security and separation of duties. Names information security and compliance practices. Result of a series of conference workshops with 50 experts.
[32]	B	Book Chapter, describes a detailed DevOps implementation approach in a Mexican data center with focus on development, quality assurance and infrastructure technology.
[11]	C	Follow up paper to [32]. DevOps approach is extended with a special focus on risk management. Written in Spanish.
[13]	C	Framework for managing needs, compliance and trust in DevOps through use of metrics and tools, processes and culture. Claims to have been validated "on a small scale in several DoD organizations" but no further elaboration.
[22]	C	Examine the impact that two IEC/ISO standards about regulated medical device software development have on DevOps practices. Theoretical study.
[23]	J	Sequel to [22]. Proposes tools and practices for DevOps to meet regulatory requirements based on the findings of the previous paper. No validation.

PAPER	TYPE	DESCRIPTION
[28]	C	Propose hard controls for development and operations in order to remain compliant with common security standards (PCI DSS, US NIST) and in control using DevOps. No empirical research.
[30]	C	Literature review on the current state of SecDevOps research.
[33]	C	Design Research. Propose a framework for successfully implementing DevOps and sharing knowledge. Validated in a small IT service firm and a large financial services company.
[35]	W	Describes how ITIL and CMMI frameworks work together with DevOps. Based on expert opinion (author).
[40]	W	Mentions critical security controls in DevOps. Based on expert opinion (author).
[43]	W	Describes how to implement DevSecOps. Based on expert opinion (author).
[53]	C	Describes general IT Governance structures found in DevOps teams. Qualitative research among team members in six companies.
[56]	C	Extended abstract of paper on difficulty of implementing DevOps in highly regulated environments and suggestions to implement secure DevOps. Only expert opinion (author) evident.

B.2 CONTROLS MENTIONED IN LITERATURE

Table B.2: Controls mentioned in literature

CONTROL	MENTIONED BY
Change control	
Automated change controls and thresholds	[9]
Version control	[13, 33, 40]
IAM & separation of duties	
Automate production deployment	[9]
Separate accounts for accessing development and productions environment	[9]
Temporary access on request for developers to production environment	[9, 28]
Code peer reviews	[9, 28, 6]
Secure Authentication	[28]
Identity and access Management	[32, 43]
Compliance	
Regular auditing	[13]
Item tracking	[23]
Standard templates in tools	[23]
Automated compliance testing and reporting	[13]
Isolation of testing and development system from production	[28, 32]
Security	
Static code analysis	[9, 6]
Automated security tests	[9, 28, 30, 40, 43, 6]
Configuration management	[13, 40, 43]
Inventory management	[40, 43]
Separation of application and databases	[32]
Monitoring and logging	
Logging	[9, 28, 43]
Continuous monitoring	[13, 33]
Reporting	[13]
Other	
BlueGreenDeployment	[9]
Risk analysis (threat model)	[11, 28]
Backup policies	[32]

Note: This table only sums up the "hard controls" as described in Section 4.5.

C CASE STUDY INTERVIEWS: CODING

Table C.1 gives an overview of all codes and code categories created in Atlas ti during the analysis of case study evidence. Since Atlas ti does not support subgroups, these were added later.

Abbreviations:
Gen. risk mitig. mechanisms: General risk mitigation mechanisms
IAM & SOD: Identity and Access Management & Separation of Duties
Mon. & Log.: Monitoring & Logging

Table C.1: Overview of codes and categories created during case study analysis

CODE	CODE GROUPS	CODE SUBGROUPS *(if applicable)*
Access to production	Risks	Team
Prioritization of Tasks	Risks	Team
Internal Fraud	Risks	Team
Increasing backlog	Risks	Team
Fast Decision Taking	Risks	Team
Access to pipeline	Risks	Team
Not working according to plan	Risks	Team
Too much team autonomy	Risks	Team
Predictability	Risks	Project
Difficulties in bigger projects	Risks	Project
Transitional Risks	Risks	Transition
Incompatibility with non-DevOps teams and processes	Risks	Organization
Security	Risks	Product
System continuity	Risks	Product
Risk of non-compliance	Risks	Product
Deploying products that are still in development	Risks	Product
Flexibility	Success Indicators	-
Less stress	Success Indicators	-
Less mistakes	Success Indicators	-
Increased rate of deployment	Success Indicators	-
Time saving	Success Indicators	-
Improved customer service	Success Indicators	-

CODE	CODE GROUPS	CODE SUBGROUPS
Consideration	Success Indicators	-
Reusable Coding	Success Indicators	-
Legacy Systems	Inhibitors	-
Scarce resources	Inhibitors	-
Compliance Requirements	Compliance Requirements	-
Frameworks	Gen. risk mitig. mechanisms	-
Quality Assurance	Gen. risk mitig. mechanisms	-
Agile Practices	Gen. risk mitig. mechanisms	-
Product Owner	Gen. risk mitig. mechanisms	-
Management Support	Gen. risk mitig. mechanisms	-
Project Management	Gen. risk mitig. mechanisms	-
Culture	Gen. risk mitig. mechanisms	-
Automation of data management	DevOps practices	-
General automation	DevOps practices	-
Continuous Improvement	DevOps practices	-
Architecture	DevOps practices	-
Continuous Deployment	DevOps practices	-
Status of DevOps Implementation	DevOps practices	-
Separation between Dev and Ops	DevOps practices	-
Cross functional teams	DevOps practices	-
Team responsible for specific service	DevOps practices	-
Test Automation	DevOps practices	-
	Controls	Change Control
Infrastructure Automation	DevOps practices	-
	Controls	Mon. & Log.
Automated Change Checks	Controls	Change Control
Limiting traffic to test change	Controls	Change Control
Change registration	Controls	Change Control
Change Advisory Board	Controls	Change Control

CODE	CODE GROUPS	CODE SUBGROUPS
Frequent Changes	Controls	Change Control
Definitions for Change Categories	Controls	Change Control
Change Authorization	Controls	Change Control
Chain Alignment	Controls	Change Control
Version Control	Controls	Change Control
		Mon. & Log.
Third Party	Controls	Security
		Change Control
Team feels responsible for their work	Controls	Security
		Change Control
		Soft Controls
Communication	Controls	Change Control
		Soft Controls
Traceability and transparency	Controls	Compliance
Auditing	Controls	Compliance
Internal Audit	Controls	Compliance
Timed Passwords	Controls	IAM & SOD
Separate roles and access rights	Controls	IAM & SOD
No access to production	Controls	IAM & SOD
Separation of Duties	Controls	IAM & SOD
Peer Reviews	Controls	IAM & SOD
Metrics	Controls	Mon. & Log.
Reporting	Controls	Mon. & Log.
Logging	Controls	Mon. & Log.
Security department	Controls	Security
		Mon. & Log.
Monitoring:rule-based alarms	Controls	Security
		Mon. & Log.
Monitoring	Controls	Security
		Mon. & Log.
Rollout & Rollback	Controls	Others
Risk Log	Controls	Others
Risk Evaluation Sessions	Controls	Others
Security Design	Controls	Security
Security Tests	Controls	Security
Security audit (internal or external)	Controls	Security
Security awareness and policies	Controls	Security

CODE	CODE GROUPS	CODE SUBGROUPS
Team is assigned responsibility	Controls	Soft Controls
Clearly communicated responsibilities, authorizations and procedures	Controls	Soft Controls
Varying maturity levels per team	*No category assigned*	-
Drivers for DevOps	*No category assigned*	-
Tooling	*No category assigned*	-
Division Infrastructure and DevOps	*No category assigned*	-
Workarounds	*No category assigned*	-
Incident handling	*No category assigned*	-

D SYNTHESIS OF LITERATURE AND CASE STUDY FINDINGS

Table D.1: Practices found in literature and case studies

CONTROL	LITERATURE	PRACTICE
General risk mitigation mechanisms		
Product owner		x
Agile practices	x	x
Quality assurance	x	x
Project management		x
Frameworks	x[1]	x[1]
Culture	x	x
Management support		x
Change control		
Automated change controls and thresholds	x	(x)[2]
Change size categories	(x)[3]	x
Version control	x	x
Change registration	(x)[4]	x
Change advisory boards		x
Automated functional testing	x[5]	x
Post-deployment approval		x
Identity and access management		
Automate production deployment	x	x
Separate accounts for accessing development and productions environment	x	x
Temporary access on request for developers to production environment	x	x
Code peer reviews	x	x
Authorization before deployment		x
Secure authentication	x	x
Access management	x	x
Access rights based on employee role		x
Compliance		
Regular auditing	x	x
Item tracking	x	x
Standard templates in tools	x	x

CONTROL	LITERATURE	PRACTICE
Automated compliance testing and reporting	x	
Isolation of testing and development system from production	x	x
Security		
Awareness trainings		x
Security department (monitoring and reaction)		x
Static code analysis	x	x
Automated security tests	x	x
Configuration management	x	x
Inventory management	x	
Security operations center		x
Separation of application and databases	x	x
Monitoring and logging		
Logging	x	x
Continuous monitoring	x	x
Reporting	x	x
Other		
Rollback strategies (e.g. BlueGreen Deployment)	x	x
Risk analysis (threat model)	x	
Risk evaluation sessions		x
Risk log		(x)[6]
Backup policies	x	x
Soft aspects		
Clearly communicated responsibilities	x	x
Team autonomy		x
Communication	x	x

1 Frameworks in literature: CMMi, ITIL, OCTAVE Allegro, OWASP SAMM, ISO/IEC 2005 Frameworks in practice: DASA, ISO 27001/2, SDL, NIST, ITIL, COBIT, SAFe

2 Implemented partly at WebSales2 immediately before deployment

3 Michener and Clager [28] mention that small changes can be deployed directly but suggest no categories

4 Implied in item tracking and logging

5 Implied in definition of Continuous Deployment/DevOps

6 Only at SmartIndustries in large projects

COLOPHON

This document was typeset using the LaTeX editor Overleaf V2 and is based on the `classicthesis` template developed by André Miede and Ivo Pletikosić.