International Journal of Accounting Information Systems 45 (2022) 100560

Contents lists available at ScienceDirect

International Journal of Accounting Information Systems

journal homepage: www.elsevier.com/locate/accinf

Rethinking IT governance: Designing a framework for mitigating risk and fostering internal control in a DevOps environment

Olivia H. Plant [a,b,*], Jos van Hillegersberg [a,c], Adina Aldea [a,d]

[a] University of Twente, Industrial Engineering & Business Information Systems, Drienerlolaan 5, Enschede, The Netherlands
[b] Quint Nederland B.V., De Oude Molen 1, Amstelveen, The Netherlands
[c] Tilburg University/ JADS, Sint Janssingel 92, Den Bosch, The Netherlands
[d] LeanIX, Prins Bernhardplein 200, Amsterdam, The Netherlands

ARTICLE INFO

Keywords:
DevOps
Risk management
Internal control
IT audit
Agile

ABSTRACT

An increasing amount of companies is transforming their IT departments towards cross-functional teams which are responsible for both development and operation of software and use automation to speed up their delivery process. This novel approach, which is commonly known as *"DevOps"*, promises many benefits such as increased speed and frequency of deployment. However, companies using DevOps are often struggling with demonstrating control of their software delivery processes to IT auditing parties, due to the decentralized decision-making structures and high degree of automation in DevOps teams. The research at hand presents a framework which aims to provide guidance to organizations in mitigating and governing risks in IT teams and departments that make use of the DevOps paradigm. We have adopted a design science research approach, building on a literature review and semi-structured interviews with seventeen employees from nine Dutch companies that are in different stages of their DevOps transition. The results suggest that two main factors which influence how departments design their DevOps environment are *risk appetite* and the *DevOps maturity*. We furthermore find that companies in practice often use a mixture of traditional, manual IT controls and the automated controls suggested in literature. Based on these insights, a situational control framework is designed which suggests suitable risk mitigation practices.

1. Introduction

The word DevOps is often used as an umbrella term to describe agile software development approaches with the aim of increasing the pace of software development processes and improving software quality (Erich et al., 2017). Important practices frequently found in DevOps teams are the shared responsibility for software development and operations and at least partially automated software delivery pipelines and infrastructure.

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 (Lichtenberger, 2017) and is nowadays continuously gaining popularity (Wiedemann et al., 2020). While many companies are enthusiastic about the opportunities that DevOps promises and are keen to implement it, they often

* Corresponding author at: University of Twente, Industrial Engineering & Business Information Systems, Drienerlolaan 5, Enschede, The Netherlands.
E-mail addresses: o.h.plant@utwente.nl (O.H. Plant), j.vanhillegersberg@utwente.nl (J. van Hillegersberg), a.i.aldea@utwente.nl (A. Aldea).

https://doi.org/10.1016/j.accinf.2022.100560
Received 3 January 2021; Received in revised form 13 January 2022; Accepted 15 March 2022
Available online 9 April 2022

worry about maintaining compliance towards regulatory organs and integrating control frameworks such as COBIT or COSO in an adequate way without hindering the efficiency of their software delivery processes. The reason for this is that high degrees of automation as well as the required decentralized decision-making structures in DevOps teams appear to contradict traditional notions of internal control and audit compliance. It therefore seems beneficial to adopt a more risk-management based approach when designing the DevOps environment, which is tailored to the specific ecosystem of a company.

A second struggle for companies as well as for their auditors is to demonstrate the control of DevOps delivery processes during IT audits by creating valid audit trails. According to DeLuccia IV et al. (2015), the increasing shift towards DevOps leads to more tension between IT and audit functions. This is especially problematic when dealing with critical software applications which support financial reporting processes that are assessed during financial statement audits. A frequently raised concern among auditing parties is the seeming lack of a separation of duty (SoD) policy in DevOps (DeLuccia IV et al., 2015). Since DevOps teams are responsible for developmental and operational processes, they require control over the entire systems development life cycle in which they are able to both write and deploy software changes on demand.

Despite the growing need for more rigorous investigations of these problems, academic research is only recently picking up on the DevOps trend. Furthermore, much of the available literature focuses on DevOps from a resource- and performance-driven perspective but neglects risk-management based considerations. IT auditing functions however stated that Agile software development approaches like DevOps call for new ways of conducting IT Audits (DeLuccia IV et al., 2015), which warrants a deeper investigation of this phenomenon in Accounting Information Systems (AIS) research. Research on this phenomenon is therefore expected to contribute significantly to the substantial body of knowledge on internal control in the AIS domain (Kumar et al., 2020).

The research at hand aims at setting a first step towards more structured risk management and process design in DevOps. Our goal is to provide a framework which guides organizations using DevOps in governing risks and providing assurance towards auditors and stakeholders on their software delivery processes, while remaining as agile as possible. To achieve this, we have adopted a design science research approach which is led by following research question:

> How can software delivery organizations ensure and demonstrate the effectiveness of their internal control environment while using the DevOps approach?

This research question encompasses two fundamental requirements which the framework needs to satisfy: Firstly, the effectiveness of the internal control environment needs to be ensured which means that suitable controls have to be identified to achieve a stable and reliable control environment. Furthermore, these controls need to be demonstrable which gives organizations the ability to show the effectiveness of their control environment to external parties such as IT auditors. Demonstration of an effective control environment may happen by enhancing traceability and visibility of IT controls, e.g. through documentation of appropriate processes. We therefore use the words ”ensure” and ”demonstrate” to respectively refer to the effectiveness and demonstrability of suitable IT controls.

Next to these fundamental requirements, the envisioned framework needs to satisfy a set of objectives in order to be of use to researchers and practitioners. Firstly, the proposed framework should obstruct the agility of the DevOps approach as little as possible. Secondly, the proposed risk mitigation and control activities in the framework should allow the companies using the framework to effectively mitigate risk as well as create valid audit trails. Lastly, the framework should be applicable to a broad set of organizations. The research at hand does therefore not intend to provide a detailed list with an exhaustive set of controls but rather aims to investigate and suggest different approaches to managing and controlling risks while using the DevOps approach. Consequently, this work has the primary intention of guiding companies in building a robust internal control environment in a DevOps context. Although we argue that the effectiveness of internal control should be demonstrable to IT auditing parties, the framework presented in this paper is therefore also relevant to companies who do not undergo financial statement or IT audits.

According to COSO (Committee of Sponsoring Organizations of the Treadway Commission, 2013), internal control is *”designed to provide reasonable assurance regarding the achievement of objectives relating to operations, reporting, and compliance”*. The proposed framework will therefore only focus on mitigating risks related to these specific types of objectives. The integration of risk management with organizational strategy is addressed separately in COSO’s *Enterprise Risk Management Framework* (Committee of Sponsoring Organizations of the Treadway Commission, 2017) and is not a part of the internal control concept. Residual risks such as those related to the achievement of strategic objectives are therefore not within the scope of this work. Furthermore, this research focuses on uncovering novel ways of addressing risks in a DevOps environment and will put less emphasis on risks and controls which remain unchanged when moving from traditional development methods towards DevOps.

2. Conceptual foundations

2.1. Internal control, risk management and the role of IT auditing

The strive for internal control in many organizations is motivated by the Sarbanes–Oxley Act (SoX) which was passed in 2002 in the U.S. following various corporate scandals. SoX requires companies to regularly report on their internal control structures and procedures that ensure the integrity of their financial reporting (Tai et al., 2018; Benaroch et al., 2012). In order to comply with these requirements, companies often adopt frameworks such as COSO which help them to structure their control processes. In its *Enterprise Risk Management* (Committee of Sponsoring Organizations of the Treadway Commission, 2004) and *Internal Control* (Committee of Sponsoring Organizations of the Treadway Commission, 2013) frameworks, COSO advocates for the implementation of appropriate risk-based controls throughout the enterprise to ensure the achievement of organizational objectives and mitigate risks which could

impact their realization. These controls can then be assessed by independent auditors (Tai et al., 2018).

IT controls mitigate risk at different levels of the organization: Auditors generally distinguish between *entity-level controls*, *IT general controls* and *application controls* (IT Governance Institute, 2006). *Entity-level controls* determine the tone and culture of an organization and include items such as strategies, policies and risk assessment activities. The *IT general controls (ITGC)* are integrated into IT processes which ensure a reliable operating environment and support the application controls. A frequently used categorization of ITGC has been established by the Public Company Accounting Oversight Board (2004) (PCAOB) which names four domains of ITGC: *program development*, *program changes*, *computer operations* and *access to programs and data*. Controls focused on program development and program changes focus on two major components: Firstly, they mitigate acquisition and implementation risks of new applications, by guarding the quality and functionality of the application (IT Governance Institute, 2006). Furthermore, they ensure adequate maintenance of existing applications by enforcing controls such as change authorizations, change documentation and testing. Computer operations controls safeguard the day-to-day delivery of information services such as configuration, installation and maintenance of IT infrastructure. Controls focused on access to programs and data mitigate the risk of unauthorized or inappropriate access to systems or data by restricting the access (IT Governance Institute, 2006). Although the aforementioned categories were defined by the PCAOB, other auditing boards such as the International Auditing and Assurance Standards Board (2019) (IAASB) share a similar conceptualization of ITGC. Lastly, *application controls* can be found in applications which support critical business processes and include completeness and accuracy of the function of an application.

COSO is one of the most popular frameworks used for SoX compliance, however, one of its limitations is that it does not explicitly name any control concepts (Rubino et al., 2017). Another framework which is often used and does provide specific controls and processes is the COBIT framework (Rubino et al., 2017). Rubino et al. (2017) advocate that COBIT can be a useful internal control framework for companies which overcomes some of COSO's limitations.

Although the SoX act is focused on financial reporting processes and intends to mitigate the risk of fraud, IT controls in general are often used as safeguards to protect the confidentiality, integrity and availability of the system information (National Institute of Standards and Technology, 2010). The strive for a robust IT internal control environment should therefore by no means be motivated solely by external pressures to be compliant with standards and laws. Research in AIS has also shown that companies who report IT internal control weaknesses often have lower accounting earnings than companies with a strong IT control environment (Stoel and Muhanna, 2011). Being in control of software development and delivery-related processes should thus be an objective for every part of an organization, regardless of the nature of the application.

2.2. DevOps

DevOps is a combination of the words *development* and *operations*. 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 (Lwakatare et al., 2016a; Smeds et al., 2015). Literature reviews have shown that there is no uniform definition of DevOps (Erich et al., 2017; Lwakatare et al., 2015) although various studies have defined some general patterns that DevOps processes usually share. Lwakatare et al. (2015) defined *collaboration*, *automation*, *measurement* and *monitoring* as the four main dimensions of DevOps. In a follow-up paper they added a fifth dimension called *culture* (Lwakatare et al., 2016a).

DevOps teams have shared goals, shared incentives and shared responsibilities for development and operations (Jabbari et al., 2016). Collaboration is enforced through information sharing and broadening of team members' skillsets (Lwakatare et al., 2015). Due to this new way of working, DevOps requires a complete shift in culture. This culture is based on trust, respect and communication (Nielsen et al., 2017) and is said to be one of the most difficult parts to implement for companies when moving towards DevOps (Bierwolf et al., 2017).

In order to achieve the maximum speed of software delivery, many DevOps teams aim at automating their software delivery process by automatically testing the code once it is checked in and merging it into the main branch (Laukkarinen et al., 2017). In some teams, code is then automatically released to a staging environment or deployed to production once it has passed all appropriate testing activities (Lwakatare et al., 2016b; Fitzgerald and Stol, 2017). These practices are known respectively as *continuous integration*, *continuous delivery* and *continuous deployment* practices. According to Fitzgerald and Stol (2017), continuous delivery and continuous deployment differ in the sense that continuous delivery implies the ability to deploy software to an environment but does not involve the actual deployment to users whereas continuous deployment involves the automated release of software builds to customers. These practices are often considered to be an integral part of DevOps according to many scholars (Laukkarinen et al., 2017; Lwakatare et al., 2016b; Smeds et al., 2015).

3. Research design

Our research adheres to the design science research methodologies by Peffers et al. (2008) as well as Wieringa (2014). Design science has been described as one of two paradigms that characterize the Information Systems research discipline, next to behavioral science (Hevner et al., 2004). While the latter focuses on the development and justification of theories, design science aims at building and evaluating artifacts which are designed to meet business needs (Hevner et al., 2004) as it is the case in our research. Geerts (2011) argues that design science methodology can provide valuable contributions to AIS research but requires a detailed methodological discussion.

We use the design science research methodology (DSRM) steps by Peffers et al. (2008) for identifying the problem at hand and defining the objectives of our solution. The approach was chosen since the DSRM by Peffers et al. (2008) provides a detailed

Table 1
Application of design science research methodologies to the research at hand.

DSRM (Peffers et al., 2008)	Design Cycle (Wieringa, 2014)	Methodology	Section
Identify Problem & Motivate	Problem Investigation	Preliminary literature review	Section 1
Define Objectives of a Solution	Treatment Design	Inference	Section 1
Design & Development		Literature review	Section 2
			Section 4
		Qualitative field study	Section 5
		Framework design	Section 6
Demonstration	Treatment Validation	Expert opinions	Section 7
		Survey	
Evaluation		Discussion	Section 8
		Conclusion	Section 9
Communication	n/a	Publication	Full Paper

description of these steps. For the design and validation of the artifact we use the design cycle phases from Wieringa (2014) since this methodology provides an elaborate set of tools and guidelines which can be applied to these tasks.

The design science approach has been implemented as follows: In the introduction of this paper (Section 1), we have identified and motivated the problem as well as inferred and discussed the objectives of our anticipated solution (Peffers et al., 2008). The design of the artifact requires the investigation of available treatments (Wieringa, 2014). For this reason, the artifact was designed based on the insights which we obtained from a literature review on risk management in DevOps which is discussed in Section 4. These insights were then refined and updated with empirical observations obtained through a qualitative field study as described in Section 5. The resulting artifact is a situational framework which can be used by practitioners working in DevOps and IT auditing domains when designing their DevOps control environment. This artifact is presented in Section 6. We have validated the framework by discussing it with industry experts and showing it to a small group of field study respondents through a survey as shown in Section 7. The implications of our research are then discussed in Section 8 while Section 9 concludes this paper. Table 1 summarizes the design science research methodology steps applied in this research.

4. Literature review

The following section presents a literature review summarizing the current knowledge and approaches for managing risk and fostering internal control in a DevOps environment. When searching through the existing body of knowledge, we focused on identifying literature that dealt with the DevOps approach in combination with at least one of the following concepts: *1. compliance*, *2. governance*, *3. control* or *4. risk management*. These keywords were chosen because they frame our research question most accurately and showed promising results according to an exploratory review. In this research we conceptualize IT governance and IT control as means to mitigate risk in an AIS context. We therefore excluded literature focused on project control theory and strategic alignment. In order to obtain this overview of literature, we adopted a multivocal literature review approach following the procedures by Kitchenham (2004) and Garousi et al. (2019). The multivocal nature of the literature review was adopted due to the lack of academic research about risk management in DevOps. According to Garousi et al. (2016), it is beneficial to include so-called "grey literature" (not peer-reviewed literature) in these cases, such as practitioner reports and white papers, since this can help the authors to gain insights into state-of the art concepts that might not be mentioned in academic literature yet and may help avoiding publication bias. However, it is important to pay special attention to the quality of the papers when conducting a multivocal literature review. We therefore only included four papers from grey literature in our sample that are ranked as "first tier" (Garousi et al., 2019) with a high credibility like whitepapers and books.

From this collection, we extracted best practices and recommendations on how to govern DevOps teams and how to minimize risks in a DevOps context. In doing so, we followed the suggestions by Webster and Watson (2002) who argue that literature reviews should be concept-centric since concepts determine the organizing structure of the review. The creation of a concept matrix supports the identification and summary of these concepts (Webster and Watson, 2002).

During this process, six dominant concepts emerged which we will use as organizing structure for our review. While the controls pertaining to the concepts can overlap to some extent, we argue that these categories represent the dimensions on which managers of DevOps teams will have to focus their efforts in order to minimize risks. The core results of the literature review are summarized in Table 2.

4.1. Change control

Change control practices intend to reduce the risk of implementing changes that lead to more failures, poor processing and unreliable systems (IT Revolution, 2015). Implementing change control is often considered to be an obstacle to running an efficient DevOps process by many companies since manual code approvals block the rapid rate of change and delivery processes. However, the DevOps book publisher IT Revolution (2015) claims that many of the change approvals and verifications that are usually done manually (e.g. performance testing, security scans, verification of change sets) can also be automated by defining thresholds and

Table 2
Overview of control practices mentioned in literature.

Change control	
Automated change controls and thresholds	IT Revolution (2015)
Version control	Farroha and Farroha (2014); Nielsen et al. (2017); Robinson (2015); Morales et al. (2018)
Identity and access management & Separation of duties	
Automate production deployment	IT Revolution (2015)
Separate accounts for accessing development and productions environment	IT Revolution (2015)
Temporary access on request for developers to production environment	IT Revolution (2015); Michener and Clager (2016)
Code peer reviews	IT Revolution (2015); Michener and Clager (2016); DeLuccia IV et al. (2015)
Secure Authentication	Michener and Clager (2016)
Identity and access management	Muñoz and Díaz (2017); Shackleford (2016)
Compliance	
Regular auditing	Farroha and Farroha (2014); Mohan et al. (2018)
Item tracking	Laukkarinen et al. (2018)
Standard templates in tools	Laukkarinen et al. (2018); Lie et al. (2020)
Automated compliance testing and reporting	Farroha and Farroha (2014); Abrahams and Langerman (2018)
Isolation of test & development system from production	Michener and Clager (2016); Muñoz and Díaz (2017)
Security	
Static code analysis	IT Revolution (2015); DeLuccia IV et al. (2015)
Automated security tests	IT Revolution (2015); Michener and Clager (2016); Mohan and ben Othmane (2016); Robinson (2015); Shackleford (2016); DeLuccia IV et al. (2015); Frijns et al. (2018); Morales et al. (2018); Rahman and Williams (2016)
Security training	Frijns et al. (2018); Mohan and ben Othmane (2016); Rahman and Williams (2016)
Configuration management	Farroha and Farroha (2014); Robinson (2015); Shackleford (2016); Lie et al. (2020); Rahman and Williams (2016)
Inventory management	Robinson (2015); Shackleford (2016); Lie et al. (2020)
Separation of application and databases	Muñoz and Díaz (2017)
Risk analysis/Threat modelling	Michener and Clager (2016); Muñoz and Díaz (2017); Rahman and Williams (2016)
Monitoring and logging	
Logging	IT Revolution (2015); Michener and Clager (2016); Shackleford (2016); Mohan et al. (2018)
Continuous monitoring	Farroha and Farroha (2014); Nielsen et al. (2017)
Reporting	Farroha and Farroha (2014); Morales et al. (2018)
Soft governance	
Define roles and responsibilities	Wiedemann (2018); Muñoz and Díaz (2017)
Communication and knowledge sharing	Wiedemann (2018); Nielsen et al. (2017); Bierwolf et al. (2017); Frijns et al. (2017); Mohan et al. (2018)
Autonomous teams	Frijns et al. (2018); Wiedemann (2018)

automated controls throughout the delivery pipeline. In order to quickly roll back deployments and trace changes, companies should always integrate version control and associated version control procedures into their DevOps processes. This can be done by using version control systems such as Git or Subversion (Nielsen et al., 2017).

4.2. *Identity and access management & Separation of duties*

Secure authentication and access management are seen as essential to controlling critical systems (Shackleford, 2016; Muñoz and Díaz, 2017; Michener and Clager, 2016). A 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 team members require access rights to development and operations environments and could therefore in many cases independently deploy a change. IT Revolution (2015) notes that it can be helpful to give DevOps engineers two accounts for the different environments e.g. an account with administrator rights in the development environment and another account with restricted user level rights in the production environment. Furthermore, the deployment pipeline should be designed in such a way that no individual holds end-to-end control over the whole process. It is suggested to automate the production deployment process so no person can execute the deployment without passing the automated controls first (Mohan et al., 2018). As another means to achieve this goal, code that is checked-in should always be peer-reviewed (DeLuccia IV et al., 2015; IT Revolution, 2015; Michener and Clager, 2016).

Multi-person authorization can be implemented in case an approved developer needs access to a specified system when he needs to fulfill operational responsibilities. Access should then be authorized and granted temporarily by a third party, e.g. 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 (IT Revolution, 2015; Michener and Clager, 2016).

4.3. Compliance

Accounts in literature vary on whether DevOps is an obstacle or a benefit to achieving compliance, since this depends highly on the circumstances. Some scholars argue that the DevOps process needs to be customized or has to be molded into a hybrid environment in order to remain compliant in specific highly regulated environments (Michener and Clager, 2016; Morales et al., 2018). However, others argue that DevOps practices help standardize processes and increase traceability by documenting processes in version management and issue tracking software (Laukkarinen et al., 2017; Lie et al., 2020). This documentation, as well as event reports detailing access to production servers form valuable input for IT audits (Michener and Clager, 2016).

Multiple practices have been suggested in literature to achieve compliance objectives while automating the deployment process as much as possible. Farroha and Farroha (2014) stress the importance of enforcing regular audits to discover irregularities early. Applications that automatically test for and report compliance violations should be integrated into the process (Abrahams and Langerman, 2018). They should terminate access if a threshold is exceeded and initiate alarms if a policy is not accepted (Farroha and Farroha, 2014).

Furthermore, many regulations demand that software items can be traced back to the requirements based on which they were developed. Laukkarinen et al. (2018) therefore propose to introduce item tracking from requirement to the final product as a standard practice in DevOps. In order to achieve further compliance, tools should include standard templates that comply with regulations.

4.4. Security

The need for a tighter integration of security measures with the DevOps paradigm has led to a new approach which is commonly known as DevSecOps or SecDevOps. The control that is mentioned most often in this category are automated security tests (DeLuccia IV et al., 2015; IT Revolution, 2015; Michener and Clager, 2016; Mohan and ben Othmane, 2016; Robinson, 2015; Shackleford, 2016; Mohan et al., 2018; Rahman and Williams, 2016) which have to be integrated into the deployment pipeline. The testing phase of the software deployment process should therefore include security tests like penetration testing, and network testing and scanning (Michener and Clager, 2016). As a general rule, all threats meeting the security criteria must be fixed or mitigated before deployment can take place (Michener and Clager, 2016). Hardware and software on servers should have secure configurations by using configuration management services that help rolling out configurations of operating systems and application components to all systems and keeping them in sync (Robinson, 2015). Robinson (2015) also mentions container technologies like Docker as a secure way of ensuring correct configurations in testing environments.

It is suggested that companies should keep an inventory of authorized and unauthorized devices and software in order to ensure platform security, version- and software management (Robinson, 2015). 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. The security department should furthermore provide regular trainings to the DevOps teams (Frijns et al., 2018; Mohan and ben Othmane, 2016; Rahman and Williams, 2016).

4.5. Monitoring and logging

Multiple papers mention the integration of process monitoring tools into the deployment pipeline (Nielsen et al., 2017; Farroha and Farroha, 2014). These tools can help minimize risk and create reliable reporting which may be used by auditors. Therefore, tools should not only monitor and automatically report incidents such as compliance breaches (Farroha and Farroha, 2014) but should also continuously perform logging to create traceable processes and valid audit trails. In case of a problem or if compliance conditions are not fulfilled, these tools can halt the deployment process and alert the developers (Farroha and Farroha, 2014).

4.6. Soft governance

Next to the previously explored traditional control categories, literature has continuously emphasized the importance of the cultural and collaborative dimension of DevOps in order to effectively govern DevOps teams (Frijns et al., 2018; Bierwolf et al., 2017). Frijns et al. (2018) advocate for a holistic approach to embed security in Agile and DevOps contexts which takes into account both hard controls (content and process) as well as soft controls (culture and relations).

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. However, DevOps culture can also potentially decrease operational risks due to the close collaboration of development and operations. 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 (Bierwolf et al., 2017).

In summary, the literature review has suggested that many traditional, manual IT controls can be automated and thus be easily integrated into the software delivery pipeline. However, the automation tools need to be embedded into appropriate procedures and policies in order to provide sufficient assurance on the control environment. Furthermore, the available literature emphasizes the importance of soft governance mechanisms in DevOps environments which are seen as a prerequisite to deploying automation activities. Lastly, the literature review has also made evident that this research domain lacks mature empirical evidence since many of the encountered papers were solely based on expert opinions or whitepapers.

5. Qualitative field study

5.1. Research methodology

In order to verify and extend the results from the literature review with empirical evidence, we have conducted a qualitative field study, including employees from 9 Dutch companies that use DevOps and have analyzed their processes and best practices related to DevOps and management of risk. Qualitative research strategies like ours emphasize an inductive approach and are commonly used for theory building (Bryman, 2012). A qualitative field study was therefore deemed to be a suitable method for gathering insights into this novel research topic which would then allow us to design a guiding framework for practitioners.

5.1.1. Data collection

Our data was collected through multiple sources:

- *Semi-structured interviews* with 17 employees of 9 Dutch organizations in 12 interview sessions. All candidates were working with DevOps in their company. We included interviewees with operational-level views such as developers, as well as managers and IT auditors with strategy-level views in our sample. An overview of all interviewees is shown in Table 3.
- *Additional documentation*: In some cases additional documentation was used for a deeper understanding of the organization and organizational processes.
- *Observations*: Five visits included a guided tour through the company. Observations were written down by the researcher after the visit and included in the analysis.

When choosing appropriate interviewees, we applied theoretical sampling which is suitable for theory building (Glaser and Strauss, 2017). In this approach, the researchers collects, analyzes and codes data until theoretical saturation is reached and no new data emerges or the theory is well developed. We took care to include a wide range of different companies and participants with differing experiences of DevOps, in order to receive diverse input and satisfy the requirement that the resulting framework should be applicable to a large range of companies. While some of the participating companies had only recently started their DevOps transition and employed some pilot teams, we also talked to representatives whose companies had been using DevOps on a mature level for multiple years.

Interviewees were found via industry contacts as well as by searching the web for suitable organizations and emailing managers from these companies. When preparing and conducting the interviews, we followed suggestions by Myers and Newman (2007) as well as Bryman (2012). The interviewees were guaranteed anonymity to ensure their trust and openness. We started each interview by introducing ourselves and explaining our research rationale. We then asked the participants to introduce themselves and elaborate on their affinity and experience with DevOps. During the first part of the interviews, we asked open questions related to the setup of the organization, the DevOps teams and their software delivery process. These topics had been prepared beforehand in an interview guide. During the second part, we showed the literature review results to the interviewees and discussed how the different control concepts were addressed in their organization. This was done at the end of the interview in order to ensure unbiased answers during the first part. After the interview, the respondents were sent the transcript on which they could provide feedback.

Table 3
Overview of companies and interviewees that were included in the study.

Company sector	Nr. of employees	Interviewees	DevOps implementation degree
Finance	>15,000 worldwide	Consultant	Piloting with several teams throughout the organization
High-Tech manufacturing	>80,000 worldwide	Manager Scrum Master	Department
Public infrastructure provisioning	>2,500	Solution Architect Product Owner DevOps Engineer	Department
Insurance	2,500	Manager Software Development Manager Infrastructure	Running a pilot team and preparing to transition the IT department
Public	>60,000	Project Manager QA Officer	Department
Public	1,500	Internal Auditor Software Engineer Product Owner Security Manager	Multiple IT teams in the organization
E-Commerce	4,500	Security Engineer	Full company
E-Commerce	>15,000 worldwide	Risk & Compliance Officer	Full company
Internet services	>100	DevOps Engineer	Full company

5.1.2. Analysis of evidence

Interviews were recorded with permission of the respondents and were written out after the interview, aiming to be as complete as possible. These transcripts and some additional documents were then analyzed with the qualitative data analysis software ATLAS.ti, following the approaches suggested by Bryman (2012) and Miles and Huberman (1994). As a first step, we applied the *open coding* technique to the transcripts by assigning paragraphs and sentences one or multiple codes that summarized the topics discussed in this section. Simultaneously, we also followed the approach of *deductive coding* (Miles and Huberman, 1994) by using the controls derived from the literature review as a starting point for creating codes. During and after the open coding process, the codes were rearranged, in some cases merged or deleted if they were redundant and relationships between codes were analyzed. The codes were then classified into categories. This process is known as *axial coding* (Bryman, 2012). Ultimately, during the *selective coding* step, we identified the core categories of our research and related the other categories to this. We found that the organizing structure which we created in our literature review also formed the core categories of our empirical evidence. We therefore use the following categories to present our findings in Section 5.2: *Change control*, *Identity and access management & Separation of duties*, *Security*, *Compliance* and *Monitoring and logging*. Furthermore, we found that the categories which predominantly influenced the nature of these core processes were the *DevOps practices* and the *compliance requirements*. This observation is discussed further in Section 6.

5.2. Results

In the following, we will elaborate on the practical implementation of the identified control processes in more detail. As mentioned earlier, the core categories of our empirical analysis are the same categories as identified during the literature review. However, soft governance mechanisms were found to be an implicit part of the above mentioned processes.

5.2.1. Change control

Respondents found change control to be an important element of the internal control environment, yet the transition towards DevOps allowed them to make some improvements to this process. Most respondents stressed that the overall amount of documentation had decreased since using DevOps and that changes could be brought into production faster. This was possible because changes were often handled by the team independently and lengthy approval processes could be avoided. For example, many companies used a change approval process in which the product owner approved a deployment before bringing it into production.

Changes were often split into low impact changes that could be deployed by the teams themselves and higher impact changes that had to be approved by independent units such as a change advisory board (CAB) first. CABs were generally observed to be present in large companies that were only partially using DevOps. A CAB could take various roles: In some companies the person committing the change could decide individually whether a CAB was necessary and could then consult it for advice. In other organizations the CAB took a more traditional change authorization role. However, since changes in code were often deployed by the DevOps teams independently instead of being approved by multiple instances, many interviewees stressed the importance of communication and team responsibility in the context of change control. I.e. an interviewee working in a financial institution emphasized that once a team deploys a change that might impact the system of another team, the team in question needs to communicate this to the team beforehand.

A frequently encountered control was the use of fully or partially automated tests that were run on every piece of code to be deployed. These tests ensured the quality of the release. In some companies, changes still had to be registered by the developers, e.g. by submitting a change request form and afterwards saving the documentation to an internal system. Furthermore, all interviewees stated to use version control on their code to track changes. Some companies also automated part of their infrastructure and found it useful that infrastructure code was versioned in the same way as application code. One respondent stated that the e-commerce company they work for used to deploy changes to their website directly with minimal testing and rerouted a small percentage of incoming web traffic to this change until a more senior developer approved the change. The approved code change was then extended to all of the incoming traffic. Nevertheless, the company had recently started to implement more preventive controls before the deployment and was aiming at making these part of the automated deployment flow.

5.2.2. Identity and access management & Separation of duties

Most considerations concerning access management concerned the question who should gain access to the production environment. The individual solutions which companies chose often depended on the level of maturity but also on the type of system which the team was dealing with. One company that had completely automated their deployment pipeline generally did not give developers access to the production environment which means that all changes had to pass the automated pipeline first and were then automatically deployed. Furthermore, changes had to be approved by two developers before they entered the pipeline. The interviewed security engineer regarded this as a solution which was both safe and efficient.

Other organizations needed to give developers access to production servers in order to enable troubleshooting. One respondent acknowledged that this level of access formed a residual risk but stressed that their organization had strong monitoring controls on the servers in order to compensate for this. Some companies also used a middle-way between these two stances and gave developers access via timed passwords which they had to request if this was necessary. All respondents stressed that they wanted to maintain a separation of duties principle in which at least two people have to approve code before it is deployed to production. Some companies enforced this technically e.g. by requiring developers to create merge requests that need to be approved before code can enter the deployment pipeline.

In general, it seemed more bothersome for companies who were still in the transition phase to deal with a separation of duties

policy than for companies who were already using DevOps at a mature level. One respondent who was working for a financial institution explained why they wanted to use timed passwords: *"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"*.

5.2.3. Security

On an organizational level, DevOps teams were often supported by the security department which frequently conducted additional monitoring activities and reacted to threats. Other security departments took more of an advisory role and conducted regular trainings for the teams. One interviewed security officer was also responsible for partly auditing the teams. Multiple companies made use of automated security tests and two companies had a dedicated security operations center that supported the teams through monitoring. Two companies had outsourced their platform to a third party that also conducted monitoring activities.

Companies again stressed the importance of team responsibility in the context of security. Since teams are ultimately responsible for security of their application, they needed to be responsible enough to come forward in case of questions or concerns. This also puts the responsibility on software organizations to create a culture in which employees dare to speak up. As a security engineer noted: *"the ugliest thing from a security point of view is if people don't dare to speak up"*.

The level of security measures employed differed per company and sector. While some companies viewed security controls as prerequisites for their processes, other companies were more flexible in their approach. One respondent working in an e-commerce company explained: *"Security is always a consideration between usability and security, the more usability the less security and the other way round"*.

5.2.4. Compliance

Most respondents indicated that they did not consider compliance to be a goal in itself but rather viewed it as a basic condition from which they needed to design their operational processes. 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 way of working. One respondent working in an e-commerce company even mentioned that they were now able to handle concerns which the auditors had faster due to working DevOps.

Multiple companies employed risk & compliance officers or internal auditors to ensure and assess compliance. One respondent noted that their company had stricter requirements towards the deployment of applications that are subject to SoX requirements than applications that were deemed less critical. Regarding our findings from the literature review which suggested to automate testing of compliance requirements, a security manager pointed out that it was generally more difficult to automate compliance controls such as data privacy checks than security controls. While it is possible to test whether information has been encrypted, most controls related to privacy of information are more difficult to automate.

A few respondents mentioned that they conducted risk evaluation sessions in their company to see whether all important risks were sufficiently addressed. Another aspect that was often mentioned in order to ensure compliance was to increase transparency and traceability of actions. This was mostly done by logging actions which is explained in the following section.

5.2.5. Monitoring and logging

Monitoring and logging was seen as very important by our respondents in order to trace back the actions that were performed on a system and to create valid audit trails. A Risk & Compliance Officer pointed out that detective controls such as monitoring and logging needed to be emphasized in DevOps to compensate for the fact that team members had a lot of autonomy and access rights: *"This is something very natural in DevOps that they [the developers] will also do the troubleshooting and deployment and everything. […] 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."*

Some respondents furthermore pointed out that the teams were innately encouraged to monitor their systems since this would help them maintain the system better and detect problems early. One company used the monitoring software Splunk for monitoring operations and had defined rule-based alarms that would send automated emails to the team's inbox as soon as an irregularity was detected. A specific role in the team was appointed the responsibility to regularly check this inbox and the Splunk dashboard.

As mentioned in the beginning of this section, soft governance mechanisms were discussed by our respondents in the context of the above mentioned control categories. They are therefore integrated into the established categorizations and do not form a separate class.

5.2.6. Synthesis of empirical insights and literature review

A comparison of risk management practices in literature and practice shows that extant literature proposes many technically advanced, automated controls while the empirical evidence suggests that many companies in practice still rely on more traditional practices. Our field study showed that the automated controls were only found in technologically mature companies and that they were not applicable or feasible for companies who were beginning their transformation or did not want to rely on complete automation of their processes. The ways in which the organizations implemented the different control categories therefore differed widely and were heavily dependent on the individual firm characteristics. Furthermore, controls from literature were mainly focused around the software delivery processes and were less concerned with supporting mechanisms such as the roles of the security department or CAB.

6. Framework

As demonstrated in the previous section, the interviewed organizations implemented the identified control processes in very different ways. While some companies had automated substantial parts of their processes, others still relied on predominantly manual controls. Organizations also differed in whether they emphasized preventive controls like deployment authorization procedures and testing or detective controls such as monitoring and logging. Even within the same company, DevOps teams often exhibited diverging approaches to risk management and internal control.

During the selective coding step of the qualitative data analysis, it became clear that the two code categories which predominantly influence the nature of these central control processes are the *compliance requirements* to which a company is subdued as well as the *DevOps practices* which it implemented. This insight led to the constatation that any prescriptive control framework needs to adopt a situational approach when addressing risk management practices and internal control measures in a DevOps environment. We extended these two above mentioned factors to broader concepts which 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*. We define these two factors as follows:

- *Risk appetite* is defined by the teams' choice whether it wants to carry the consequences which the occurrence of common risks such as a faulty deployment will have on the company. A high risk impact as well as many compliance requirements often lead to a lower risk appetite. However, risk appetite was also found to vary per team within the same company. This dimension is thus heavily based on team culture and should rather be seen as a matter of choice instead of being entirely defined by external constraints.
- *DevOps maturity* is defined as the extent to which teams are able to operate independently as well as to what extent they are capable of automating their software deployment process. A high DevOps maturity therefore requires teams to exhibit both technical and soft skills such as frequent communication and assumption of responsibility for the end-to-end process.

Mapping these factors on two axes results in the matrix shown in Fig. 1 with four basic situations in which teams can find themselves. We will call this matrix the *DevOps risk management matrix*.

We then assigned the control practices encountered in literature and practice to the quadrants of the matrix which led to the creation of four risk mitigation strategies that seem suitable for each situation. These four strategies are described in the following sections and are summarized in Fig 2 along the four domains of ITGC as defined by the PCAOB (Public Company Accounting Oversight Board, 2004). Due to the similarities of controls, we have combined the program development and program changes domains. Additionally, we have added a section describing the recommended focus of soft governance activities for the high maturity columns. This section is not included in the lower half of the matrix, since the suggested mechanisms there rely on rather traditional controls whereas soft governance becomes more critical in later stages of the transformation journey where the teams work more independently.

The control categories indicate which types of risk the strategies can mitigate and how organizations should address these risks when growing in maturity: Controls addressing program development and changes primarily mitigate the risk of deploying faulty software to the production environment. The suggested controls are therefore preventive in nature. On the other hand, the proposed controls related to computer operations are detective and aim to uncover unusual behavior in the system such as faulty programs or

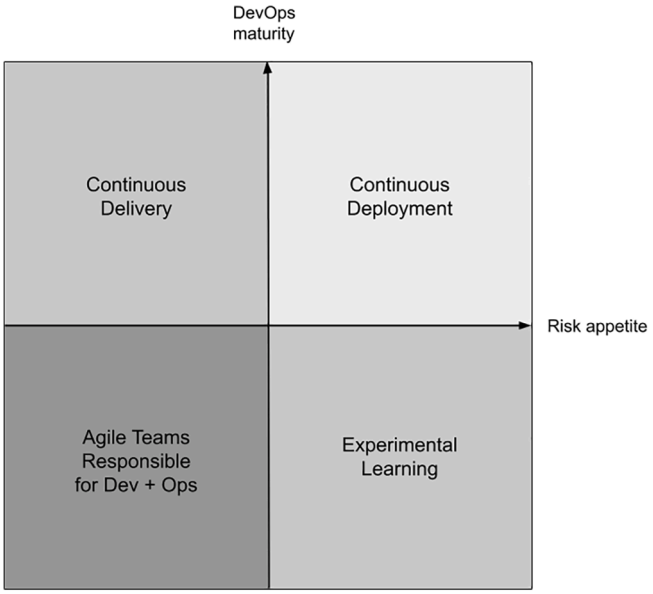

Fig. 1. DevOps risk management matrix.

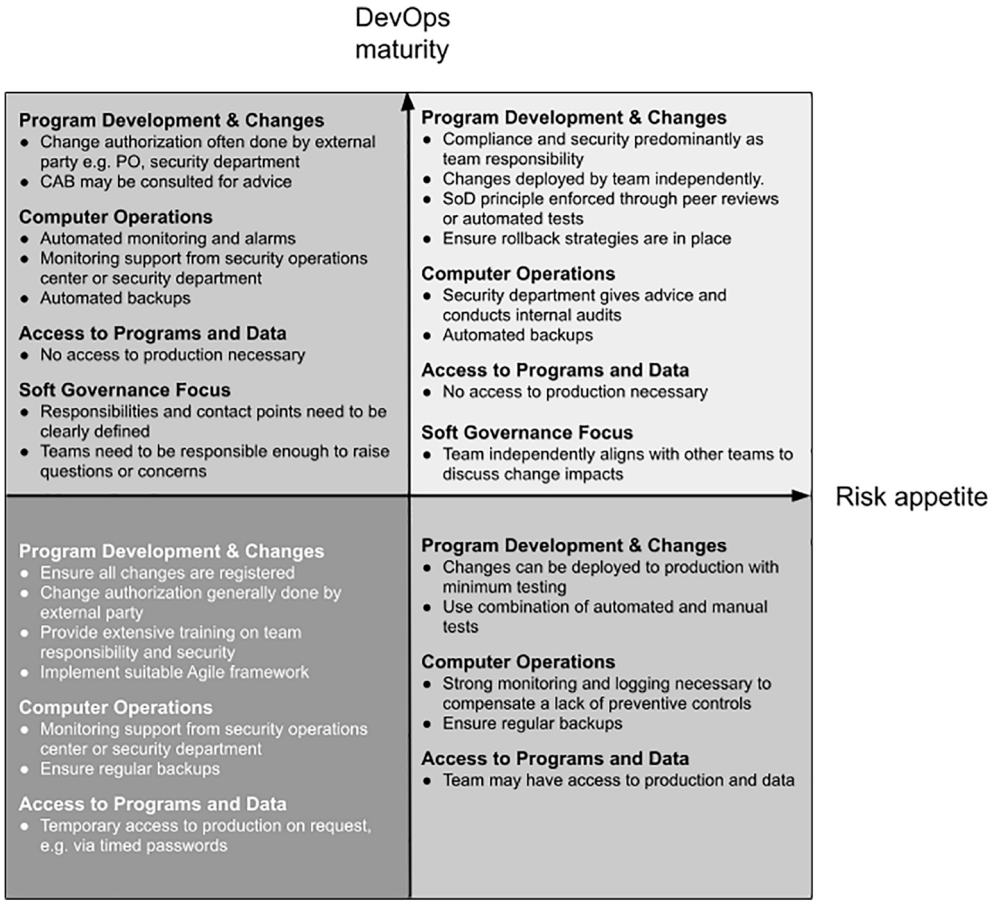

Fig. 2. Summary of suggested control strategies on the DevOps Risk Management Matrix.

malicious traffic as soon as possible. These controls are geared towards ensuring that systems run stable and can be recovered quickly if necessary. Lastly, the strategies regarding access to programs and data ensure that only authorized employees may access the production environment. This is done to prevent users from conducting changes which may harm the system or receive information which they are not authorized to obtain. The only notable exception to this is the "experimental learning" strategy in which we argue that teams may have full access to the production server since the team and the application are able to accept and support the consequences of the aforementioned risk.

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 darker colours low agility. 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. While teams with a higher risk appetite and a high maturity can fully automate their process, take on complete responsibility for their system and thus achieve the maximum possible speed and agility, the same degree of automation is not suggested for teams with a low risk appetite, even if they are very mature in their processes. This is due to their preference for employing additional preventive, manual controls that ensure compliance and risk mitigation before deployment.

6.1. 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. For this reason, the strategy presented in this quadrant is based on the continuous delivery principle: 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 always be required right before deployment. This can for example be done by the product owner of the team. In some cases, a CAB may be involved in the deployment of large scale changes. Separating applications as much as possible from each other, for example by using a microservices architecture and employing container technology, helps to limit the impact of software changes and thus minimizes risk. Generally, developers do not

require access to production environments since the deployment happens automatically after the manual approval.

Despite the large amount of preventive and detective controls in this strategy, it is very important for organizations 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. Teams need to be responsible enough to reach out to the security department or the CAB in case they have questions or concerns instead of just pushing critical changes to production. On a general note, since approval in a continuous delivery setting can often be given with one single click on the button, this strategy does not lose much of its speed compared to a continuous deployment approach, if the responsible person authorizes important changes right away.

6.2. 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 code to production servers. Most importantly, the strategy presented in this quadrant resembles the rather idealistic definitions of DevOps encountered during the literature review the most. 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. The automated logging of deployment authorizations and versioning of code ensures the creation of documentation which may be used for IT audits. Since the deployment is fully automated, it is not necessary to give all developers access to production which lowers the risk of internal fraud. The security department takes an advisory role which the teams can approach if they have questions and audits the processes in regular intervals. Furthermore, teams should always have a strategy in place which allows them to quickly roll back a faulty change and conduct strong monitoring activities that detect and escalate incidents quickly. Since the DevOps teams deploy changes independently, it is important to establish close collaboration between the teams so that releases which might impact each other's systems are communicated in a timely manner.

6.3. 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 minor 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 approving a change after is has already been deployed to the production server 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 lower the possibility of deploying faulty code. Teams in this quadrant are also more likely to give developers access to production servers, although companies will then have to implement strong monitoring and logging instruments to ensure traceability of actions. 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.

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

Teams that are still relatively immature in using DevOps and cannot or do not want to take many risks are operating in a rather perilous environment. Therefore, they have to be very careful when implementing new features and methods. A suitable starting point is to implement an existing Agile framework to structure their processes and to give teams the responsibility for both development and operations. Automation does not yet play a large 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 maturity growth process. Changes are often still approved by a CAB, except for very small changes with minimal impact, and should always be registered beforehand to improve traceability and inform other teams. Teams 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.

7. Validation of IT artifact

Design artifact validations can be artificial or naturalistic in nature and may be undertaken with the intention to either improve the artifact (formative evaluation) or to judge the extent to which the artifact matches the expectation (summative evaluation) (Venable et al., 2016). Within the research at hand, we have chosen to apply purely artificial evaluation strategies since they are suited to prove or disprove the design theory and the utility of the artifact. The initial stages of the validation process were formative whereas the feedback obtained at the end of the process was utilized to understand the final contribution of the framework.

7.1. Interviews with industry experts

The DevOps risk management framework was validated formatively by discussing it with four industry experts. According to Wieringa (2014), expert opinions are a useful research methodology when evaluating artifacts since the experts can imagine how the

Table 4
Experts interviewed for artifact validation.

Interviewee	Expertise	Experience
Senior Manager (E.1)	Governance, Risk & Compliance of Technology, IT Audit, DevOps	9 years
Senior Manager (E.2)	Agile, DevOps	11 years
Senior Consultant (E.3)	Agile, DevOps, Change Management	10 years
Director (E.4)	IT Audit, IT Risk	12 years

artifact will work once it is implemented in practice. The participants were selected based on their affinity with either DevOps, IT Risk Management or IT Auditing, as well as due to their broad experience and independent position with regards to our research. All of the experts worked in consultancy or auditing functions and had encountered a wide range of organizations over the course of their careers. Furthermore, the experts were not part of any of the nine companies studied during the field study and none of the experts was priory involved in our research. An overview of the interviewed experts and their background is provided in Table 4.

During the interviews, the experts were presented with the DevOps risk management matrix. We firstly discussed the validity of the matrix in itself, asking for their opinion whether the two factors that constitute the matrix were indeed the best aspects to consider when designing situational control strategies. We then discussed the separate strategies which we had designed and the corresponding controls with the experts and asked them for their opinion and potential improvements.

The industry experts overall agreed with the risk management matrix and its respective strategies. Some experts pointed out that the two axes in the matrix were not entirely independent since 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.

E.3 disagreed with the more traditional control strategies such as those mentioned in the low maturity/low risk appetite quadrant. This expert furthermore argued that companies in high risk environments could completely automate their deployment pipeline instead of using the suggested continuous delivery approach since this would ensure that all code is tested thoroughly before it is deployed to the production server. On the other hand, E.4, who had a background in IT auditing and IT risk, thought that complete automation of the deployment process 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.

We furthermore discussed the implications of auditing the DevOps approach with E.1 and E.4. When asked about the importance of soft governance controls in Agile and DevOps, E.4 stated that they believed 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. This importance creates a significant responsibility for the learning & development department of organizations to ensure that employees are trained properly and are continuously made aware of their responsibilities and the new processes. The same expert acknowledged that the shift towards DevOps 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 require the design of new ITGC to ensure that they are always up to date.

The expert interviews led to some incremental changes such as renaming the horizontal axis of the framework from *"risk impact"* to *"risk appetite"*. Furthermore, some controls pertaining to the different strategies were slightly adjusted or added. [1]

7.2. Survey sent to field study participants

The framework was also sent to the interviewees that had participated in our qualitative field study who could then give feedback on the expected effectiveness and efficiency of the artifact in their own company. The interviewees were sent an online survey presenting the near-final version of the framework. The survey consisted of open and closed questions that could be ranked on a five-point scale. The main evaluands to be validated during this stage were the *usefulness* of the artifact as well as the *effectiveness* which means that the respondents were asked to rate aspects such as the effectiveness of the proposed strategies in terms of risk mitigation and to what extent they would hinder an efficient DevOps process. No questions were obligatory and respondents were asked to only give feedback where they deemed necessary. The survey was answered by six employees from at least four different companies. The exact number of companies participating cannot be determined since not all respondents chose to disclose their identity.

Similar to the interviewed experts, the respondents had varying opinions about the use of traditional controls in a DevOps environment. Some respondents pointed out that automation of the deployment process would be beneficial in a high-risk environment and lead to more frequent testing while another respondent was concerned that the traditional controls would undermine the autonomy of DevOps teams.

After careful consideration, we decided to hold on to the proposed continuous delivery strategy for organizations with a low risk appetite, based on the considerations made by E.4 who was an expert in IT Auditing and IT Risk. The control strategies were designed in such as way that the DevOps teams remain largely autonomous in their decision-making process, even in companies with a lower risk appetite.

The vast majority of respondents (four out of six) agreed that the proposed strategies and controls would address the most important risks whereby only one respondent was neutral and one disagreed. Similarly, four respondents thought that the proposed

[1] Please note that the framework which we present in Section 6 is the final version of the framework and includes these changes.

controls would not obstruct the DevOps way of working unnecessarily while two respondents were neutral.

Overall, the matrix was evaluated positively on its risk mitigation ability and very positively on the agility requirement. Due to the situational nature of the DevOps risk management matrix, the artifact is furthermore applicable to a large set of companies. We therefore conclude that the artifact sufficiently satisfies the objectives which were defined at the beginning of this research.

8. Discussion

8.1. Implications of findings

In this study, we have designed a framework that provides guidance to organizations using DevOps in designing their internal control environments and creating valid audit trails.

To this end, we have reviewed and synthesized existing literature and conducted a qualitative field study interviewing 17 employees from a range of 9 different companies. The insights were ultimately used to design a situational framework which can be used by AIS practitioners and researchers when designing and assessing internal control environments in a DevOps context. The framework has been validated by four experts with backgrounds in DevOps, IT Risk and IT Auditing, as well as by showing them to a small group of field study participants.

Our research shows that the traditional domains of IT control and their underlying risks remain relevant in a DevOps context. However, the nature of the employed IT controls is fundamentally different to traditional development contexts. The DevOps approach inherently emphasizes detective control mechanisms such as monitoring applications and infrastructure and logging actions (Lwakatare et al., 2016a; Ghantous and Gill, 2017). Literature has suggested that many preventive controls may be automated in DevOps which allows companies to perform quality and security assurance on their software while maintaining high speed and frequency of deployment (Mohan et al., 2018; IT Revolution, 2015). Contrary to these expectations created in literature, our empirical investigation has found that most companies are not yet ready to automate their software delivery processes and IT controls completely, either because they are still in the transition phase or because they have a low risk appetite and do not want to rely on complete automation of their processes. The automated control mechanisms found during our literature review were therefore often not suitable for the companies we interviewed. Our empirical research has led to the novel insight that many DevOps organizations have designed a hybrid environment where they employ a mixture of manual and automated controls. Furthermore, some of our interviewees even stated that they do not intend to move towards completely automated releases of software since they did not deem this necessary for their purpose. This contradicts the implicit assumption of extant literature in which a successful DevOps implementation is often equated with automated deployment of software. It therefore seems crucial that IT risk managers adopt a situational approach when designing the internal control environment of DevOps teams and consider the specific maturity and risk appetite of the team in question. Our risk management framework is meant to provide guidance in this process.

Our findings support previous research which has defined culture to be an indispensable part of DevOps (Erich et al., 2017; Lwakatare et al., 2015). Moreover, we highlighted the fact that cultural mechanisms are paramount for establishing an effective internal control environment which is in line with previous literature on DevOps security (Frijns et al., 2017; Mohan and ben Othmane, 2016; Bierwolf et al., 2017). It has become evident that soft controls like communication and the assumption of team responsibility function as a safeguard for many of the other IT controls.

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. Employees need to be encouraged to step forward if they have made a mistake, knowing they will not be punished for it. Furthermore, companies need to ensure that teams communicate sufficiently among each other and are aware of their responsibilities. Proposals on how to foster knowledge sharing practices in a DevOps context have been made by Nielsen et al. (2017) and Hemon et al. (2020).

DevOps emphasizes shared responsibility and ownership (Lwakatare et al., 2019). In addition to this, our interviewees found it important to clearly assign and communicate these responsibilities in the organization, 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. In this we also see a shift of responsibility from what is traditionally considered to be the ”second line of defense” towards the DevOps teams. Departments like security and risk management now take on more of an advisory role towards DevOps teams whereas the responsibility for the correct execution of these processes lies within the team.

Despite the challenges encountered, all interviewees were positive about transitioning towards DevOps, suggesting that the DevOps approach yields some advantages over traditional software development methods. The risk management framework has therefore been designed in such a way that it supports and accompanies the teams throughout their maturity growth. As we have discussed in this section, this transition brings about some challenges related to risk management and control. However, our literature review has shown that automation of software delivery can improve compliance and internal control significantly compared to traditional development methods through the use of automated logging and monitoring tools as well as automated infrastructure. These tools ensure the creation of valid audit trails and ensure correct configuration of the development and production environments at all times. Our interview results suggest that it becomes easier to manage the control environment as more parts of the pipeline are automated while hybrid environments are more difficult to navigate.The fact that many companies from our field study however currently apply hybrid approaches emphasizes the necessity to remain vigilant and engage in a continuous risk dialogue as suggested by Bierwolf et al. (2017).

8.2. Contributions to literature and practice

Our results provide novel contributions to both AIS research and practice. To the best of our knowledge, this study presents the most holistic research about risk management and internal control in DevOps to this date. Previous research has focused on subsets of our topic such as integrating security (Frijns et al., 2018; Mohan et al., 2018; Rahman and Williams, 2016) or compliance practices (Farroha and Farroha, 2014; Abrahams and Langerman, 2018) into the DevOps pipeline. Furthermore, Wiedemann et al. (2019) have created a framework on how to combine project and operations control in DevOps teams to achieve alignment with organizational objectives.

Extant literature had a very idealized view on managing risks in DevOps, suggesting many automated processes which were only encountered in few companies during this study. The automated controls in literature often did not address the problems that most companies were struggling with. We have therefore extended the current body of knowledge with novel insights into how DevOps software delivery organizations design and manage their control environment with a hybrid approach.

From a practitioner perspective, our framework provides guidance to IT Auditors, IT Risk Managers and DevOps organizations on how to design and assess the internal control environment and manage risk in a DevOps context.

8.3. Limitations and further research

The results of this study should be viewed in the light of their limitations. Firstly, we have only talked to representatives of companies located in the Netherlands. Further research is necessary to investigate whether our findings can be generalized to other cultural environments. Furthermore, our literature review has shown that most of the extant literature is based on qualitative research approaches. We therefore call for more quantitative approaches investigating the impact of the DevOps approach on organizational operations and risk management.

An ongoing challenge for practitioners is the application of the soft governance mechanisms which were addressed in this study. Future research is necessary on how to implement these controls as well as how to document them so they could potentially be considered in DevOps audit and assurance engagements in the future.

Lastly, our risk management framework has only been validated artificially through expert opinions. In future research, the framework could also be tested in a naturalistic setting by applying it to a case study and quantifying the results.

9. Conclusion

This study aimed at designing a framework for organizations to manage risks and increase internal control while using the DevOps approach. Secondly, it intended to research how these companies can demonstrate their internal control towards IT auditing parties.

It is argued that there is no best way to implement DevOps and that the DevOps concept rather needs to be tailored to the situation of the company in question. Two main factors that influence companies in their decision how to manage their control processes are the *DevOps maturity* and *risk appetite*. Based on these factors, the situational framework is developed that suggests four strategies with suitable controls to manage risks in DevOps.

Furthermore, 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, IT auditors will have to find a way to assess these soft governance mechanisms in order to reliably give assurance on internal control in DevOps environments.

Declaration of competing interest

In accordance with Elsevier policy and our ethical obligation as a researchers we report that this study was partially conducted as part of a research internship at KPMG Netherlands. Furthermore, the first author has been employed at the consultancy company Quint Nederland. Neither company has provided any specification regarding the content of this research.

Acknowledgments

We would like to thank KPMG Netherlands and Quint Nederland B.V. for their interest and support while conducting this research. We would also like to thank the seventeen participants of the field study and the four experts who have helped us validate our results. Lastly, we are grateful to the two anonymous reviewers and the Associate Editor for their valuable suggestions in improving this research.

References

Abrahams, M.Z., Langerman, J.J., 2018. Compliance at velocity within a devops environment, in: 13th International Conference on Digital Information Management, ICDIM 2018, Berlin, Germany. pp. 94–101. DOI: 10.1109/ICDIM.2018.8847007.

Benaroch, M., Chernobai, A., Goldstein, J., 2012. An internal control perspective on the market value consequences of IT operational risk events. Int. J. Accounting Inform. Syst. 13, 357–381. https://doi.org/10.1016/j.accinf.2012.03.001.

Bierwolf, R., Frijns, P., van Kemenade, P., 2017. Project management in a dynamic environment: Balancing stakeholders, in: IEEE European Technology and Engineering Management Summit, E-TEMS 2017, Munich, Germany. pp. 1–6. DOI: 10.1109/E-TEMS.2017.8244226.

Bryman, A., 2012. Social research methods, 4th ed. Oxford University Press, Oxford.

Committee of Sponsoring Organizations of the Treadway Commission, 2004. Enterprise risk management - Integrated framework. https://www.coso.org/Pages/erm-integratedframework.aspx.

Committee of Sponsoring Organizations of the Treadway Commission, 2013. Internal control - Integrated framework. https://www.coso.org/Documents/990025P-Executive-Summary-final-may20.pdf.

Committee of Sponsoring Organizations of the Treadway Commission, 2017. Enterprise risk management - Integrating with strategy and performance.

DeLuccia IV, J., Gallimore, J., Kim, G., Miller, B., 2015. DevOps audit defense toolkit. Technical Report. IT Revolution. https://itrevolution.com/devops-audit-defense-toolkit/.

Erich, F.M.A., Amrit, C., Daneva, M., 2017. A qualitative study of DevOps usage in practice. J. Softw.: Evol. Process 29. https://doi.org/10.1002/smr.1885.

Farroha, B.S., Farroha, D.L., 2014. A framework for managing mission needs, compliance, and trust in the DevOps environment. In: IEEE Military Communications Conference MILCOM, Baltimore, MD, USA, pp. 288–293. https://doi.org/10.1109/MILCOM.2014.54.

Fitzgerald, B., Stol, K.J., 2017. Continuous software engineering: A roadmap and agenda. J. Syst. Softw. 123, 176–189. https://doi.org/10.1016/j.jss.2015.06.063.

Frijns, P., Bierwolf, R., Zijderhand, T., 2018. Reframing security in contemporary software development life cycle, in: IEEE International Conference on Technology Management, Operations and Decisions, ICTMOD 2018, IEEE, Marrakech, Morocco. pp. 230–236. DOI: 10.1109/ITMC.2018.8691277.

Frijns, P., van Leeuwen, F., Bierwolf, R., 2017. Risk management – from risk log to risk dialogue, in: International Conference on Engineering, Technology and Innovation, ICE/ITMC 2017, IEEE, Madeira, Portugal. pp. 699–703. DOI: 10.1109/ICE.2017.8279953.

Garousi, V., Felderer, M., Mäntylä, M.V., 2016. The need for multivocal literature reviews in software engineering. In: 20th International Conference on Evaluation and Assessment in Software Engineering, EASE. ACM, New York, NY, USA, pp. 1–6. https://doi.org/10.1145/2915970.2916008.

Garousi, V., Felderer, M., Mäntylä, M.V., 2019. Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. Inf. Softw. Technol. 106, 101–121. https://doi.org/10.1016/j.infsof.2018.09.006.

Geerts, G.L., 2011. A design science research methodology and its application to accounting information systems research. Int. J. Accounting Inform. Syst. 12, 142–151. https://doi.org/10.1016/j.accinf.2011.02.004.

Ghantous, G.B., Gill, A.Q., 2017. DevOps: Concepts, practices, tools, benefits and challenges, in: 21st Pacific Asia Conference on Information Systems, PACIS 2017, Langkawi Island, Malaysia.

Glaser, B.G., Strauss, A.L., 2017. The discovery of grounded theory. Routledge. https://doi.org/10.4324/9780203793206.

Hemon, A., Fitzgerald, B., Lyonnet, B., Rowe, F., 2020. Innovative practices for knowledge sharing in large-scale DevOps. IEEE Softw. 37, 30–37. https://doi.org/10.1109/MS.2019.2958900.

Hevner, A.R., March, S.T., Park, J., Ram, S., 2004. Design science in IS research. MIS Quarterly 28, 75–105.

International Auditing and Assurance Standards Board, 2019. International standard on auditing 315: Identifying and assessing the risks of material misstatement. https://www.ifac.org/system/files/publications/files/ISA-315-Full-Standard-and-Conforming-Amendments-2019-.pdf.

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

IT Revolution, 2015. An unlikely union: DevOps and audit - Information security and compliance practices. Technical Report. DevOps Enterprise Forum. https://itrevolution.com/book/devops-and-audit/.

Jabbari, R., Bin Ali, N., Petersen, K., Tanveer, B., 2016. What is DevOps?. In: Proceedings of the Scientific Workshop Proceedings of XP 2016. ACM, New York, NY, USA, pp. 1–11. https://doi.org/10.1145/2962695.2962707.

Kitchenham, B., 2004. Procedures for performing systematic reviews. Technical Report. Software Engineering Group, Department of Computer Science. Keele University.

Kumar, S., Marrone, M., Liu, Q., Pandey, N., 2020. Twenty years of the International Journal of Accounting Information Systems: A bibliometric analysis. Int. J. Accounting Inform. Syst. https://doi.org/10.1016/j.accinf.2020.100488.

Laukkarinen, T., Kuusinen, K., Mikkonen, T., 2017. DevOps in regulated software development: Case medical devices. In: IEEE/ACM 39th International Conference on Software Engineering: New Ideas and Emerging Technologies Results Track (ICSE-NIER). IEEE, Buenos Aires, Argentina, pp. 15–18. https://doi.org/10.1109/ICSE-NIER.2017.20.

Laukkarinen, T., Kuusinen, K., Mikkonen, T., 2018. Regulated software meets DevOps. Inf. Softw. Technol. 97, 176–178. https://doi.org/10.1016/j.infsof.2018.01.011.

Lichtenberger, A., 2017. Fünf kritische erfolgsfaktoren für eine erfolgreiche DevOps transformation [Five critical DevOps success factors]. HMD Praxis der Wirtschaftsinformatik 54, 244–250. https://doi.org/10.1365/s40702-017-0293-6.

Lie, M.F., Sanchez-Gordon, M., Colomo-Palacios, R., 2020. DevOps in an ISO 13485 regulated environment: A multivocal literature review, in: 14th ACM/ IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM 2020. DOI: 10.1145/3382494.3410679.

Lwakatare, L.E., Kilamo, T., Karvonen, T., Sauvola, T., Heikkilä, V., Itkonen, J., Kuvaja, P., Mikkonen, T., Oivo, M., Lassenius, C., 2019. DevOps in practice: A multiple case study of five companies. Inf. Softw. Technol. 114, 217–230. https://doi.org/10.1016/j.infsof.2019.06.010.

Lwakatare, L.E., Kuvaja, P., Oivo, M., 2015. Dimensions of DevOps. In: Lassenius, C., Dingsøyr, T., Paasivaara, M. (Eds.), Agile processes in software engineering and Extreme Programming. Springer International Publishing, Cham, pp. 212–217.

Lwakatare, L.E., Kuvaja, P., Oivo, M., 2016a. An exploratory study of DevOps: Extending the dimensions of DevOps with practices. In: 11th International Conference on Software Engineering Advances, ICSEA 2016, Rome, Italy, pp. 91–99.

Lwakatare, L.E., Kuvaja, P., Oivo, M., 2016b. Relationship of DevOps to Agile, Lean and Continuous Deployment. In: Abrahamsson, P., Jedlitschka, A., Nguyen Duc, A., Felderer, M., Amasaki, S., Mikkonen, T. (Eds.), Product-focused software process improvement. Springer International Publishing, Cham, pp. 399–415.

Michener, J.R., Clager, A.T., 2016. Mitigating an oxymoron: Compliance in a DevOps environment. In: IEEE 40th Annual Computer Software and Applications Conference, Atlanta, GA, USA, pp. 396–398. https://doi.org/10.1109/COMPSAC.2016.155.

Miles, M.B., Huberman, A.M., 1994. Qualitative data analysis: An expanded sourcebook, 2nd ed.,. SAGE.

Mohan, V., ben Othmane, L., 2016. SecDevOps: Is it a marketing buzzword? Mapping research on security in DevOps, in: 11th International Conference on Availability, Reliability and Security, ARES 2016, Salzburg, Austria. pp. 542–547. 10.1109/ARES.2016.92.

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

Morales, J.A., Yasar, H., Volkman, A., 2018. Implementing DevOps practices in highly regulated environments. In: 19th International Conference on Agile Software Development: Companion, XP'18. ACM, New York, NY, USA, pp. 1–9, 10.1145/3234152.3234188.

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

Myers, M.D., Newman, M., 2007. The qualitative interview in IS research: Examining the craft. Inf. Organ. 17, 2–26. https://doi.org/10.1016/j.infoandorg.2006.11.001.

National Institute of Standards and Technology, 2010. Contingency planning for information systems: Updated guide for federal organizations. Technical Report ITL bulletin for July 2010. U.S. Department of Commerce. http://ws680.nist.gov/publication/get_pdf.cfm?pub_id=906210.

Nielsen, P.A., Winkler, T.J., Nørbjerg, J., 2017. Closing the IT development-operations gap: The DevOps knowledge sharing framework, in: Joint Proceedings of the BIR 2017 pre-BIR Forum, Workshops and Doctoral Consortium.

Peffers, K., Tuunanen, T., Rothenberger, M.A., Chatterjee, S., 2008. A design science research methodology for information systems research. J. Manage. Inform. Syst. 24, 45–77. https://doi.org/10.2753/MIS0742-1222240302.

Public Company Accounting Oversight Board. Auditing standard no. 2: An audit of internal control over financial reporting performed in conjunction with an audit of financial statements. https://pcaobus.org/oversight/standards/auditing-standards/details/Auditing_Standard_2.

Rahman, A., Williams, L., 2016. Software security in DevOps: Synthesizing practitioners' perceptions and practices, in: International Workshop on Continuous Software Evolution and Delivery, CSED 2016, Raleigh, NC, United States. pp. 70–76. DOI: 10.1145/2896941.2896946.

Robinson, A., 2015. Continuous security: Implementing the critical controls in a DevOps environment. Technical Report. SANS Institute. https://www.sans.org/reading-room/whitepapers/critical/continuous-security-implementing-critical-controls-devops-environment-36552.

Rubino, M., Vitolla, F., Garzoni, A., 2017. The impact of an IT governance framework on the internal control environment. Records Manage. J. 27, 19–41. https://doi.org/10.1108/RMJ-03-2016-0007.

Shackleford, D., 2016. A DevSecOps playbook. Technical Report. SANS Institute. https://www.sans.org/reading-room/whitepapers/analyst/devsecops-playbook-36792.

Smeds, J., Nybom, K., Porres, I., 2015. DevOps: A definition and perceived adoption impediments. In: Lassenius, C., Dingsøyr, T., Paasivaara, M. (Eds.), Agile processes in software engineering and Extreme Programming. Springer International Publishing, Cham, pp. 166–177.

Stoel, M.D., Muhanna, W.A., 2011. IT internal control weaknesses and firm performance: An organizational liability lens. Int. J. Accounting Inform. Syst. 12, 280–304. https://doi.org/10.1016/j.accinf.2011.06.001.

Tai, V.W., Lai, Y.H., Yang, T.H., 2018. The role of the board and the audit committee in corporate risk management. North Am. J. Econ. Finance. https://doi.org/10.1016/j.najef.2018.11.008.

Venable, J., Pries-Heje, J., Baskerville, R., 2016. FEDS: A framework for evaluation in design science research. Eur. J. Inform. Syst. 25, 77–89. https://doi.org/10.1057/ejis.2014.36.

Webster, J., Watson, R.T., 2002. Analyzing the past to prepare for the future: Writing a literature review. MIS Quarterly 26 xiii – xxiii. 10.1.1.104.6570.

Wiedemann, A., 2018. IT governance mechanisms for DevOps teams - How incumbent companies achieve competitive advantages, in: 51st Hawaii International Conference on System Sciences, HICSS 2018, Waikoloa Village, Hawaii, USA. pp. 4931–4940. 10.24251/HICSS.2018.617.

Wiedemann, A., Wiesche, M., Gewald, H., Krcmar, H., 2020. Understanding how DevOps aligns development and operations: A tripartite model of intra-IT alignment. Eur. J. Inform. Syst. 1–16 https://doi.org/10.1080/0960085X.2020.1782277.

Wiedemann, A., Wiesche, M., Thatcher, J.B., Gewald, H., 2019. A control-alignment model for product orientation in Devops teams-a multinational case study. In: 40th International Conference on Information Systems, ICIS 2019, Munich, Germany, pp. 1–17.

Wieringa, R.J., 2014. Design science methodology for information systems and software engineering, 1st ed.,. Springer-Verlag, Berlin Heidelberg. https://doi.org/10.1007/978-3-662-43839-8.