European Journal of Information Systems

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/tjis20

Understanding how DevOps aligns development and operations: a tripartite model of intra-IT alignment

Anna Wiedemann, Manuel Wiesche, Heiko Gewald & Helmut Krcmar

To cite this article: Anna Wiedemann, Manuel Wiesche, Heiko Gewald & Helmut Krcmar (2020) Understanding how DevOps aligns development and operations: a tripartite model of intra-IT alignment, European Journal of Information Systems, 29:5, 458-473, DOI: 10.1080/0960085X.2020.1782277

To link to this article: https://doi.org/10.1080/0960085X.2020.1782277

EUROPEAN JOURNAL OF INFORMATION SYSTEMS
2020, VOL. 29, NO. 5, 458–473
https://doi.org/10.1080/0960085X.2020.1782277

THE OPERATIONAL RESEARCH SOCIETY
Taylor & Francis
Taylor & Francis Group

EMPIRICAL RESEARCH

Understanding how DevOps aligns development and operations: a tripartite model of intra-IT alignment

Anna Wiedemann [a], Manuel Wiesche [b], Heiko Gewald [a] and Helmut Krcmar [c]

[a]Neu-Ulm University of Applied Sciences, Neu-Ulm, Germany; [b]Chair for Digital Transformation, TU Dortmund University, Dortmund, Germany; [c]Chair for Information Systems, Technical University of Munich, Munich, Germany

ABSTRACT

A top priority of organisations around the globe is to achieve IT-business alignment at all levels of the organisation. This paper addresses operational alignment within IT functions. Traditionally, IT functions are divided into highly independent subunits. In the face of pressure to adapt to rapidly changing customer demands and to manage increasingly complex IT architectures, many organisations have begun implementing joint, cross-functional DevOps teams, which integrate tasks, knowledge and skills pertaining to planning, building, and running software product activities. In this study, we examine eight cases of DevOps implementation. We apply grounded theory to identify three mechanisms comprising a tripartite model of intra-IT alignment: individual componentization, integrated responsibility, and multidisciplinary knowledge. Our model provides insights into how alignment between development and operations can be achieved in DevOps teams within the IT function.

ARTICLE HISTORY
Received 15 September 2018
Accepted 10 June 2020

SPECIAL ISSUE EDITORS
Likoebe Maruping and Sabine Matook

KEYWORDS
DevOps; software development; software operation; agile; intra-IT alignment; misalignment; operational alignment

1. Introduction

IT-business alignment remains a key concern among information technology (IT) executives (Gerow et al., 2014; Reynolds & Yetton, 2015). Traditionally, information systems (IS) functions are divided into separate subunits, including software development and software operations (Hemon et al., 2018). This organisational structure hinders cross-functional collaboration and alignment across different subunits within the IT function (Constantinides & Barrett, 2014; Gregory et al., 2018; Swanson & Beath, 1989). As demands on the IS function have grown, closer cooperation between the IT subunits development and operations has proven essential to achieve agility and alignment throughout the complete software delivery lifecycle (Krancher et al., 2018).

Establishing consistency between such organisational functions to realise the full potential of information systems has been the primary focus of alignment research (Gerow et al., 2014). Much prior alignment research concentrates on the four components business strategy, IT strategy, business infrastructure and processes, and IT infrastructure and processes, focused mainly on the strategic link between business and IT (Gerow et al., 2014; Reynolds & Yetton, 2015). On the operational level, alignment between the IT subunits, which we refer to as intra-IT alignment, is equally important, since misaligned IT subunits can adversely affect overall cohesion within the IT functions and negatively impact the goals of business and IT (Onita & Dhaliwal, 2011; Wagner et al., 2014).

Recent alignment literature has identified several dimensions of misalignment in the IT function. First, as development cycles grow shorter, the development prioritises providing new software features quickly, while operations prioritises ensuring stable running systems with as few changes as possible (Edberg et al., 2012; Fitzgerald & Stol, 2017). If success is measured and standards are defined follow solely the speed logic or solely the stability logic, misalignment is likely. Second, the different goals of development and operations make it challenging to combining knowledge, communicate clearly, and deliver the best possible IT services to the organisation and its customers (Krancher et al., 2018). Little research has addressed how such intra-IT misalignment can be best resolved.

On the operational level, research has examined alignment between software development and user requirements (Ramesh et al., 2010). For instance, agile software development methods align developers and users by enhancing collaboration among them (Hemon et al., 2018; Maruping & Matook, forthcoming). However, these concepts do not focus on software operations, leaving traditional separation of development and operations IT subunits and intra-IT misalignment unresolved (Dhaliwal et al., 2011; Hemon et al., 2018). In order to understand the process of aligning internal IT development and operation subunits, this paper seeks to answer the following research question: *What are the mechanisms by which*

CONTACT Anna Wiedemann anna.wiedemann@hnu.de

development and operations IT subunits achieve intra-IT alignment?

To answer our research question, we study DevOps, a phenomenon that has gained importance in practice over the last years (Forsgren et al., 2018). The DevOps method integrates the tasks, knowledge and skills pertaining to planning, building, and running software product activities in a joint cross-functional team within the IT function (Wiedemann, Forsgren et al., 2019). We used an exploratory, multiple case study design based on 26 interviews with DevOps experts. These DevOps teams jointly plan, develop and operate IT software and architecture solutions using an integrated software delivery lifecycle and thereby bridging development and operations (Fitzgerald & Stol, 2017). We find that DevOps integrates the advantages of agile software development to react quickly to customer demands but also broadens agility to operations such as software architecture, responsibilities and knowledge (Hemon et al., 2018). We develop a grounded model with three mechanisms which facilitate intra-IT alignment through DevOps teams. We contribute to operational alignment literature by providing insights into the process of achieving development and operations alignment in the IT function (Onita & Dhaliwal, 2011). Our model shows how well-aligned IT subunits can implement agile software maintenance and shift their mindset from project orientation to product orientation.

2. Related literature

This section provides an overview of the three research streams relevant to our study. We compare agile software development and DevOps, briefly summarise the general strategic business-IT alignment research, and discuss gaps in research on operational alignment and misalignment that indicate a need for new theory.

2.1. Agile software development and DevOps

IT functions are typically divided into several subunits, often including separate units for software development and operations (Fitzgerald et al., 2006; Swanson & Beath, 1989). Traditional approaches to managing software development include top-down planning and sequential implementation (Bick et al., 2017). With separated subunits, the software operations subunit takes over once a new software component is installed and failures or problems appear (Kim & Westin, 1988; Swanson & Beath, 1989). Many organisations desire to align software development and operations to facilitate collaboration (Wiedemann, Wiesche et al., 2019).

To meet business requirements through better software development, more and more organisations are switching from traditional software development approaches to agile software development methods (Bick et al., 2017; Maruping et al., 2009). Agile software development methods provide a flexible and lightweight alternative to traditional plan-driven project management methods (Fitzgerald et al., 2006). The goals of agile methods are to increase transparency of project progress, create usable interim products and services, and respond more quickly and efficiently to new or changing customer requirements (Bick et al., 2017). The method focuses on collaboration within the software development subunit and with customers (Kude et al., 2019; Maruping & Matook, forthcoming).

The DevOps method extends agile software development by focusing not only on the development subunit, but also on the operations subunit (Wiedemann, Forsgren et al., 2019). It adds scope and speed of delivery and bridges the gap between the two silo IT subunits to form a cross-functional team (Hemon et al., 2018; Krancher et al., 2018).

By combining software developers' and software operations' perspectives into one cross-functional team (Krancher et al., 2018), firms seek to achieve intra-IT alignment, build consensus within cross-functional teams and increased levels of agility. Integrating the DevOps method between the two separated subunits of development and operations increases alignment by including their tasks in joint cross-functional teams (Hemon et al., 2018; Krancher et al., 2018).

2.2. Strategic business-IT alignment

IS research has focused extensively on alignment, which remains a top concern among IS executives (Gerow et al., 2014; Henderson & Venkatraman, 1993). Alignment is defined as *"the degree to which the needs, demands, goals, objectives, and/or structures of one component are consistent with the needs, demands, goals, objectives, and/or structures of another component"* (Nadler & Tushman, 1993, p. 119).

The Strategic Alignment Model (SAM) structures how firms align strategic choices that support realising the potential of IT (Henderson & Venkatraman, 1993). The fundamental premise of the SAM is that IT can be managed more effectively if choices made across the four domains of business strategy, IT strategy, organisational infrastructure and processes, and IT infrastructure and processes are aligned. SAM distinguishes between three categories of business-IT alignment: intellectual alignment, cross-domain alignment and operational alignment (Gerow et al., 2014; Henderson & Venkatraman, 1993).

Intellectual alignment takes place at the executive level and includes business and technology scope, competencies, and governance (Henderson & Venkatraman, 1993). Cross-domain alignment refers to the degree of fit and integration between IT strategy,

business strategy, IT infrastructure, and business infrastructure (Chan & Reich, 2007). Operational alignment applies to infrastructure, business and IT processes, focusing on cooperation within the same level of business and IT (internal) (Gerow et al., 2014), in order to align processes, skills and architectures. As outlined above, the current research focuses on operational alignment, which we will discuss in greater detail in the following.

2.3. Forms of operational alignment and misalignment

This section discusses prior research in the area of operational alignment and identifies the need to better understand intra-IT alignment. Intra-IT alignment involves closer integration of the daily work of developers and operations staff. Misalignment is most apparent in firms that too narrowly customise IT systems to meet current strategic needs, resulting in an inflexible, substandard infrastructure which is costly to update (Shpilberg et al., 2007).

Research into operational alignment focuses primarily on the relationship between business and IT (Gerow et al., 2014). Relatively little research is available on how operational alignment is achieved across subunits within the IT function (Dhaliwal et al., 2011). Following the lead of operational alignment literature, we consider intra-IT operational alignment and misalignment in terms of goals, processes, competencies, and interoperability, as summarised in Table 1 below.

From a **goals** perspective, intra-IT development and operations activities can have different scopes and pursue different aims (Fichman & Melville, 2014). Developers prioritise innovation to realise strategic agendas, whereas operations aim to provide stability with focus on daily business (Fichman & Melville, 2014; Markus & Keil, 1994). Hence, misalignment can occur when there is a lack of shared objectives.

From a **procedural** perspective, intra-IT development and operations activities have different workflows and methods (Bick et al., 2017; Kim & Westin, 1988). Developers tend to follow formal processes and apply software development methods, whereas operations people tend to work ad hoc and reactively, applying informal methods (Cram & Newell, 2016; Edberg et al., 2012). Hence, misalignment can occur when there is a lack of shared process focus.

From a **competencies** perspective, intra-IT development and operations activities have different knowledge backgrounds and skills. Developers are usually skilled in understanding strategic goals and requirements and developing suitable solutions, whereas operations staff generally skilled in solving problems and managing requests (Edberg et al., 2012). Hence, misalignment can occur when there is a lack of shared competences.

Finally, from an **interoperability** perspective, intra-IT development and operations activities are allocated differently. Whereas developers often work proactively to avoid or resolve emerging business problems or achieve a strategic goal, operations staff generally work reactively when problems appear (Edberg et al., 2012; Onita & Dhaliwal, 2011). Hence, misalignment can occur when the approaches are not combined.

In summary, we apply the four operational alignment perspectives (goals, procedures, competencies and interoperability) identified in general alignment literature to consider potential misalignment within

Table 1. Perspectives on operational alignment and forms of intra-IT misalignment.

	Description	Intra-IT Misalignment
Goals	Align operational IT goals and business value goals (Gerow et al., 2014; Rivard et al., 2006). Alignment involves achieving commitment by linking objectives across functional areas capabilities. Close alignment positively impacts performance (Bharadwaj et al., 2007; Powell, 1992).	Misaligned business and IT goals threatens cost efficiency and stability in turbulent environments (Gerow et al., 2014; Rivard et al., 2006). In intra-IT development and operations, goals of speed lead to conflict with goals of stability. A gap in research is the developing of shared intra-IT goals despite different organisational views (Edberg et al., 2012; Fichman & Melville, 2014).
Procedures	Align operational IT processes and technology with business processes to benefit customers (Barua et al., 2004). Alignment can be achieved by continuously adapting reconfiguring organisational and IT infrastructure and processes to create business value (Chan & Reich, 2007; Vermerris et al., 2014).	Misaligned intra-IT processes, infrastructure and workflows can threaten customer satisfaction (Kang et al., 2008). Developers use flexible lightweight processes (agile manifest) to achieve innovation and change (Bick et al., 2017; Tiwana & Konsynski, 2010). Operations favour stable processes with less change to avoid failures (Kim & Westin, 1988).
Competencies	Align operational IT competencies and cognitive and structural patterns with business competencies (Wagner et al., 2014). Social relations in operational alignment are based on principles of shared understanding, communication, and trust between business and IT personnel (Martin et al., 2008; Wagner et al., 2014).	Misaligned intra-IT competencies in terms of communication and knowledge exchange can threaten business value (Wagner et al., 2014). Whereas developers communicate new software functionalities und use software code to document, operations persons rely on existing documents and guidelines to solve problems The education of the two professional fields is very different (Edberg et al., 2012).
Interoperability	Achieve reciprocal effects and cooperation to facilitate congruent collaboration. Strengthening the relationship between different IT subunits improves alignment within the IT function (Dhaliwal et al., 2011; Onita & Dhaliwal, 2011).	Intra-IT operational interoperability has been studied in terms of software development and testing (Onita & Dhaliwal, 2011). A gap in research is how intra-IT interoperability alignment is achieved between different IT subunits of development and operations.

the IT function. With the notable exception of Dhaliwal et al. (2011) and Onita and Dhaliwal (2011), who focus on the alignment of development and testing, scholars have not investigated intra-IT operational alignment, and little is known about resolving operational misalignment between IT development and operations.

3. Research methodology

To fill this research gap, we adopt a qualitative research methodology, which is best suited to study novel phenomena and provide rich explanations. We conducted a multiple case study in diverse settings to gain compelling results (Yin, 2018). By analysing alignment in DevOps teams within multiple organisations, we answer our research question based on the analysis of real-life situations (Eisenhardt, 1989; Yin, 2018). The philosophical position of this qualitative method has an underlying interpretive epistemology because we strive to interpret social practices (Walsham, 1995). We collected our data in in-depth field investigation (Urquhart, 2012) and adopt a classic conceptualist-grounded theorising technique. The generalisation of our theoretical concept is extended through the inductive concepts generated by the multiple case study and extant theory, e.g., alignment literature, as recommended by Glaser and Strauss (1967).

3.1. Data collection

We collected primary and secondary data on DevOps cases from organisations in eight industries in Germany by conducting interviews and collecting additional case information. This broad scope is essential to our research method because it enables us to study a varied pattern of alignment. Our primary data stems from 26 semi-structured interviews with DevOps team members in eight IT organisations. Our secondary data draws on company reports and publications, including company blog articles (see Appendix A).

In collecting data, we followed the guidelines proposed by Sarker and Sarker (2009). We identified suitable case study participants at practitioner conferences where these organisations were presented as outstanding good examples for DevOps integration. Since our research relies on theoretical sampling (Glaser & Strauss, 1967) we selected the eight DevOps teams due to their similarities as well as differences. In theoretical sampling, relevance and purpose are essential. Regarding relevance, the selection process guarantees that the cases are theoretically useful in terms of replicating or extending theory (Eisenhardt, 1989). All cases of this research have implemented development and operations activities in cross-functional teams. Still, the cases are not identical. The DevOps teams differ in terms of organisational setting, responsible IT service, organisational conditions, industry, size, and cultural transformation (Eisenhardt, 1989). By selecting cases with different team constellations, we aimed to explore different perspectives in aligning development and operations activities.

After selecting potential cases, we contacted the firms via email, telephone, and in person to achieve a high level of credibility, explaining our research and guaranteeing anonymity to build a trusting relationship (Myers & Newman, 2007).

We identified appropriate interview partners through discussions and also based on recommendations of other interviewees following the "snowballing" method (Sarker & Sarker, 2009). We chose one DevOps team per case based on conversations with managerial and technical employees about alignment and misalignment between development and operations.

We visited five of the eight firms on-site to ensure strong contact between researcher and interviewee. Our semi-structured interviews typically lasted about an hour each. All interviews were held by one researcher of the team, either personally, via telephone, or video conference. Table 2 illustrate a brief overview of our cases and Appendix 1 depicts detailed information of the DevOps settings of the participating cases. Every interview was recorded and transcribed and we took extensive notes during the interviews. After every interview, a memo was written which included a summary of the key insights and follow-up questions for the next interview (Urquhart, 2012). We began each interview by introducing ourselves and our research, followed by questions about DevOps-related experience and current job position. The main body of the interview consisted of questions about alignment between development and operations (see questionnaire in Appendix B).

3.2. Data analysis

Following the principle of emergence of grounded theory, the categories emerged from our data (Glaser & Strauss, 1967). Following the grounded theory method (GTM), we used theoretical sampling, rigorous coding, memo writing, and constant comparison when analysing our data (Glaser, 1978). The Glaserian approach fits well with our research objective, as it allowed us to shape our research intent very broadly to shed light on the mechanisms that support alignment between IT development and operations (Wiesche et al., 2017). The Glaserian approach is particular useful in examining the causes of relationship between categories (Urquhart, 2012). In our analysis, we developed an understanding of what appeared in the data through conceptualisation based on theoretical sensitivity (Glaser, 1978). In line with

Table 2. Case description.

Case	Brief Description	Interviewees
Case 1	A leading food and convenience retail company with more than 100,000 employees. Team set up in 2014 and had been in place for 4 years when studied.	Six interviews with four team members, product owner, and agile coach
Case 2	A leading financial banking institution with more than 100,000 employees. Team set up 2016 and had been in place for 2–3 years when studied.	Three[a] interviews with former group manager, group manager, and two team members
Case 3	A leading insurance company with more than 10,000 employees. Team set up 2017 and had been in place for 1–2 years when studied.	Two[a] interviews with executive, manager, and team lead
Case 4	An Internet company with more than 1,000 employees. Team set up 2012 and had been in place for 6 years when studied.	Three interviews with director IT, team lead, and one team member
Case 5	A leading retail company with more than 50,000 employees. Team set up 2015 and had been in place for 3 years when studied.	Two interviews with team lead and team member
Case 6	A warehousing business with more than 20,000 employees. Team set up 2015 and had been in place for 3 years when studied.	Three interviews with team lead, and two team members
Case 7	A leading foods and convenience retailer with more than 100,000 employees. Team set up 2014 and had been in place for 4 years when studied.	Three[a] interviews with division manager, two managers, and one team member
Case 8	A well-known online travel agency with more than 1,000 employees. Team set up 2017 and had been in place for 1–2 years when studied.	Four interviews with team lead and three team members

[a]interviews were held with two interviewees people

GTM, we identified concepts related to resolving the misalignment of the development and operations components of IT teams (Urquhart, 2012).

After each interview, we wrote memos to synthesise new findings and identify issues (Wiesche et al., 2017). We used these memos iteratively to refine our interview questions and approach (Urquhart, 2012). We strengthened our GTM by triangulating our data with secondary data, including publicly available data such as company websites, blogs, as well as insights collected at conferences. Throughout the theory development process, we consciously suppressed our prejudices and avoided applying existing theory to our data (Birks et al., 2013). We relied on previous research during our data collection phase, recognising the value of comparing alignment literature with our own data, rather than using it to guide our research (Glaser, 1978). Ultimately, our concept of tripartite intra-IT alignment emerged through the systematic generation and conception of data (Glaser & Strauss, 1967).

We collected 577 pages of transcripts, which we coded using NVivo 9 based on 377 initial codes focusing on alignment/misalignment. We iteratively refined our concept in a cross-validation process among research team members to ensure reliability (Yin, 2018). In a first step, we started coding according to the a priori-defined misalignment perspectives (Mis-A): goals, procedures, competencies, and interoperability. We applied open coding along these areas as a method of identifying mis-/alignment in DevOps teams and to understand the nature of misalignment. Table 3 presents an overview of our open coding.

In a second step, we used selective coding to identify the mechanisms through which DevOps teams achieve alignment between development and operations. Afterwards we used theoretical sampling to develop a strong link between data collection and analysis (Birks et al., 2013; Glaser & Strauss, 1967). The resulting model describes how the intra-IT alignment mechanisms resolve misalignment within the IT function (Table 4).

Table 3. Misalignment between development and operations and open codes.

Mis-A	Open Codes
Mis-A1: Goals	• Significant effort spent managing old legacy systems • IT function still works with legacy systems managed by data centres • Dependencies to other functions within the company
Mis-A2: Procedures	• Waiting to release new software until end of sprint • Lack of willingness to adopt service responsibility • Making decisions without integrating the complete team • Formal processes for problem management • Friction losses due to different understanding and different backgrounds
Mis-A3: Competencies	• Fear of losing intellectual property • People have specialist knowledge in one area • "Finger pointing" because of failures
Mis-A4: Interoperability	• Hidden dependencies between different teams • Identification were the problem comes from (own service or other) • Planning backlog without operations or development

Table 4. Overview of tripartite intra-IT alignment.

Nature of Alignment	Misalignments	Alignment Mechanisms
Alignment of IT development operations activities in DevOps teams.	• Goals • Procedures • Competencies • Interoperability	• Individual componentization • Integrated responsibility • Multidisciplinary knowledge

Across all our cases, we found various components of our three alignment mechanisms. While each case emphasised different components, we found aspects of all three mechanisms in every team. In the following, we discuss how DevOps teams use the alignment mechanisms to address misalignment.

4. Findings

Our analysis reveals three core mechanisms used to achieve intra-IT alignment between development and operations in cross-functional teams: individual

Table 5. Three alignment mechanisms of intra-IT alignment.

Alignment mechanism	Description	Components
Individual componentization	Individual componentization refers to multi-layered architecture arrangements with a microservices architecture that serves as a limited architectural workspace. It provides a high level of authorisation through self-services for teams regarding technology selection in order to achieve individually configurable end products.	• Silent releases • Containerisation • Convertible infrastructure
Integrated responsibility	Integrated responsibility is defined as the team's accountability for managing all tasks and processes of the software delivery lifecycle. This includes planning the tasks, building new or change existing software code, running the software, and fixing failures when they appear.	• Extending agile • Process automation • Product orientation
Multidisciplinary knowledge	Multidisciplinary knowledge is defined as the new appropriation, deepening, and distribution of necessary skills and knowledge regarding plan, build and run tasks within the team. Problems are solved collaboratively.	• Competency broadening • Problem ownership • Skill distribution

componentization, integrated responsibility, and multidisciplinary knowledge (see Table 5). Our results suggest that the interplay between these three mechanisms address the different forms of misalignment identified above.

4.1. Achieving intra-IT alignment: resolutions for misalignment

Based on our theoretical findings, we present mechanisms for solving misalignment between development and operations and for achieving intra-IT alignment.

4.1.1. Building of individual componentization

Individual componentization is the mechanism by which a DevOps team creates a flexible IT architecture that supports and requires integrated responsibility and multidisciplinary knowledge to achieve intra-IT alignment. In this paragraph we explain how three components of individual componentization resolve misalignment: silent releases, containerisation and convertible infrastructure. Before establishing DevOps teams, all of the organisations we investigated had separate development and operations IT subunits. They reported the need to established new technology and replace monolithic IT architectures with modern components in order to achieve individual componentization in the DevOps teams.

One component of individual componentization is silent releases. Silent releases are defined as the continuous deployment of new software functionalities without application downtimes in the productive software operations environment. In traditional set-ups, releasing especially large new software functionalities can be problematic because the release generally requires a system outage. Silent releases enable alignment in interoperability (Mis-A4), because time-consuming dependencies with other organisational units, e.g., waiting for release weekends, are resolved by giving DevOps teams responsibility for operational tasks like software deployments. Silent releases provide new software components without system outages because releases are frequent and small. Case 1 resolves the discrepancy between development and operational Mis-A4 (interoperability) by enabling silent releases. Moving from huge legacy systems towards a software architecture that enables the DevOps team to make silent releases and deploy new software components continuously and when necessary enables DevOps teams to develop new functionalities and make release decisions to satisfying business demands quickly, silently and without complicated coordination efforts. *"You make a release but you do not tell anyone before . . . For example, in marketing we decide, we develop a campaign and from one day to the next it is online"* (Case 1, team member).

Another component of individual componentization is containerisation. Containerisation is defined as an encapsulated and interchangeable virtual operating system. Containers enable the provision of applications and tools for development activities and are maintainable by the team. A common problem in monolithic architecture is the high level of dependencies among the components. Containers do not share data with any other services without an integrated application interface (API). Containers can be shipped to the running software system. Case 2, a very large bank, invested great effort to address goals (Mis-A1) within a DevOps team. Containerisation software helped them to gain acceptance from developers as well as operations people that their different goals can be aligned in one common goal (Mis-A1). This mechanism from individual componentization allows DevOps teams to organise their work with corresponding tools and technology. For example, Cases 2, 5 and 8 work with containers to manage new software functionalities on their own desktop which can be easily set up. *"This is like the container system in shipping. I can define that I have these containers and they are always the same. Then I can automate everything [. . .] and we get this DevOps cycle"* (Case 2, team member).

Convertible infrastructure is the third component of individual componentization. Convertible infrastructure is defined as a flexible IT architecture that is tailored to and administrated by the DevOps team. The DevOps team resolves problems associated with monolithic architecture and enables other teams to work with a convertible infrastructure. Furthermore,

integrating a high level of self-services authorises IT professionals to manage their IT architecture by themselves. Following this convertible infrastructure approach, Case 1, for example, addresses **interoperability (Mis-A4) between the team members by enhancing speed, because operations activities proactively** identify failures before the customer does with the help of monitoring tools and alerts. Convertibility of IT infrastructure components enables speed and predictive operations. *"We provide a platform that allows [the team] to do their job. . . . we enable them to build software, to operate, to deploy, and to monitor it"* (Case 6, team lead).

4.1.2. Enabling of integrated responsibility

Integrated responsibility is the mechanism by which a DevOps team creates accountability for the complete software delivery lifecycle that supports and requires individual componentization and multidisciplinary knowledge to achieve intra-IT alignment. Integrated responsibility helps resolve misalignment through three components: extending agile, process automation, and product orientation. Integrated responsibility describes the end-to-end responsibility of the DevOps teams. Traditionally, several IT subunits of an IT function are necessary to provide software to the end user, including the operations IT subunit to fix problems that arise. In the DevOps model, the DevOps team is responsible for performing all activities.

The first component of integrated responsibility is extending agile. Extending agile is defined as the customisation of one or more agile methods to achieve the necessary process guidance and also individualised flexibility. The agile development method "Scrum" has two to four weekly release cycles, which is too slow for some DevOps teams. To achieving the advantages of rapid software delivery through DevOps, they can extend the agile method and shorten the release cycles. This addresses procedures (Mis-A2). During the integration of DevOps teams, Case 4 organised their work and releases within their DevOps teams using the Scrum agile method. After a while, the team was able to deploy new software features before the sprint cycles ended, so the team extended the scrum method by drawing on the best supportive mechanisms from several lean and agile methods (e.g., Scrum and kanban). DevOps members of Case 4 determined that existing agile software development methods unsatisfactorily supporting their work. "*We call it Scrumban . . . We have a product that must work on the pulse of time. Scrum is too slow for us. We must be agile, in the sense of fast. This does not mean that we do not need any processes . . . I need a certain queue with tasks. But this queue must be adaptable*" (Case 4, manager).

The second component of integrated responsibility is process automation. Process automation is the capacity to conduct necessary workflow steps automatically without removing responsibility from the team. In traditional setups, it can take a long time to make decisions between separated development and operations IT subunits. DevOps teams broaden the agile principle of continuous integration to continuous deployment or continuous delivery. DevOps teams commonly have shared responsibility for the entire software delivery lifecycle. Such teams benefit from a high degree of automation that reduces arrangements and manual steps, for example, in testing. DevOps teams in Cases 2 and 8 achieved a high degree of process automation to resolve procedures (Mis-A2) by avoiding formerly necessary successive manual working steps, e.g., releases. Before implementing DevOps, Cases 2 and 8, for example, made great effort to eliminate manual process steps: *"Continuous deployment . . . There are automated tests that have been used before. The department says: 'The following ten tests are running for this package'. If they are always successful, any development on this package can always go live. I do not have to look at it anymore"* (Case 2, team member).

The third component of integrated responsibility is product orientation. Product orientation is defined as the work structure that changes the formal work arrangement from a project involving a pre-defined end and time- and result-oriented controls and incentives, to a product-oriented arrangement that sees the software as an ongoing endeavour that requires a continuous approach to control and incentives. Traditionally, software is developed and delivered in IT projects with a defined start and end date. The end of the projects is typically combined with a software release, whereupon responsibility is transferred to operations. In DevOps set-ups, the team is responsible for the complete software delivery lifecycle and must make decisions quickly when necessary. To facilitate flexibility and distribute tasks efficiently within the DevOps teams, our findings indicate a movement from a management-led project orientation to a product orientation. The DevOps team at Case 3 is responsible for ensuring that an internal delivery platform is available to support other teams. This addresses procedure (Mis-A2) because every team member contributes to decisions made regarding the software delivery lifecycle. *"There should not be a DevOps project. We have a product and are responsible for it end-to-end. This is a never-ending project* [. . .] *and it does not have a 'D-Day'. There will always be improvements and operations and the more the platform is used, the more we have to do"* (Case 3, executive).

4.1.3. Integrating multidisciplinary knowledge

Multidisciplinary knowledge is the mechanism by which a DevOps team develops the development and

operations skillset and knowledge needed to conduct all software-relevant activities that supports and requires individual componentization and individual responsibility to achieve intra-IT alignment. Based on our results, we identified three core multidisciplinary knowledge components relevant to resolving misalignment: competency broadening, problem ownership, and skill distribution.

The first mechanism of multidisciplinary knowledge is competency broadening. In traditional setups, IT professionals specialise either in development or in operations. In DevOps teams, competency broadening is the expansion specialist knowledge of a team member to include broad knowledge in both areas. Team members with development backgrounds must obtain operational competency and be able to fix bugs, while team members with operations backgrounds must acquire development competency and engage with business processes and programming. Case 6 stressed that the members of DevOps teams need to constantly broaden their competency, moving beyond typically one-dimensional backgrounds. Competency broadening addresses misalignment in goals (Mis-A1) and competencies (Mis-A3). Our cases indicate that the team size varies between four people in Case 7 and fifteen people in Case 2. No matter how many people are in the DevOps team, they must manage all service-related tasks, from planning new requirements to developing software feature to building and running the infrastructure. Competency broadening helps to achieve a common goal between developers and operations experts because the people see the advantages of these broad competencies in combining development and operations knowledge *"We are looking for someone who says 'I can do Ops and I am interested in Dev, or I can Dev and I'm into Ops.' If the person shows willingness to learn, s/he has the job"* (Case 6, team lead).

The second mechanism of multidisciplinary knowledge is problem ownership. Problem ownership is defined as the responsibility for every team member to fix failures related to the IT services run by the DevOps team. In IT functions with several IT subunits, the subunit is only responsible for their certain tasks. Since developers and operations people in DevOps teams have shared responsibility for the software delivery lifecycle, everyone is responsible when a problem appears. Classic role concepts with strict boundaries of responsibility are less and less common. Problem ownership addresses misalignment in the area of goals (Mis-A1). The DevOps team in Case 2 follows a common goal and common management style in organising their tasks. *"At first, it was an act of trust between my colleague and myself. We talk to people . . . to find out if there is anyone who can do it better than we do? What are they doing differently? What can you learn from them?"* (Case 2, former group manager).

The third mechanism of multidisciplinary knowledge is skill distribution. Skill distribution is defined as the degree to which skills are shared and distributed within the team to guarantee high level software development and delivery. In traditional IT functions, highly specialised people in IT subunits possess certain skills. In DevOps teams, these skills need to be distributed between the team members to ensure that not every team member has to know everything. Case 8 shows that skill distribution solves misalignment in competencies (Mis-A3), through a suitable team organisation and by guaranteeing that responsibility for all necessary tasks is shared broadly within the team. The team lead from Case 6 spent lot of time and effort to find people with the complete skill needed to work in a DevOps teams, *"The first vacancy was published 18 months ago and we looked for a long time."* Case 8 of our investigation recognised these skills limits and set up a team structure where people still have dedicated roles, but immediately start acquiring the new skills needed to manage their service. *"We do not have all the skills . . . but we always support redundancies. If someone has to do something, s/he usually is really concerned with transmitting knowledge. For example, not everyone can do database administration in our team, but at least . . . we all share some basic knowledge"* (Case 8, team member).

5. Discussion

Modern software development ecosystems require high degrees of orchestration (Huber et al., 2017), as in these ecosystems, fast-changing requirements require rapid and continuous change in software applications under development (Fitzgerald & Stol, 2017). One approach to orchestrate software development processes and the existing software landscape is DevOps, the integration of tasks, knowledge and skills pertaining to planning, building, and running software product processes in a joint team within the IT function.

The DevOps method focuses on orchestrating development and operations subunits within the IT function. DevOps thereby not only implements automated processes to enable continuous development (Fitzgerald & Stol, 2017), but also builds links within the overall IT department to prevent isolated solutions. The examined DevOps teams in this study are responsible for both platform solutions and the corresponding applications, thus creating their own internal ecosystems which need to be orchestrated accordingly. We found that individual componentization, integrated responsibility, and multidisciplinary knowledge help DevOps teams coordinate their work. Thereby, DevOps ensures operational alignment within the IT function.

Our research contributes to operational alignment literature in several ways: We identify three mechanisms to achieve alignment in DevOps teams within the IT function. We integrate these into a tripartite model of intra-IT alignment for resolving misalignment. Finally, we highlight our contribution to intra-IT alignment and explain how it relates to operational alignment. In the following, we integrate our three mechanisms in a coherent model and discuss the implications for theory and practice.

5.1. Analytical summary: a model of tripartite intra-IT alignment

Based on the three emergent mechanisms individual componentization, integrated responsibility, and multidisciplinary knowledge, we develop a model of tripartite intra-IT alignment (see Figure 1). Our model contributes to prior research by extending operational alignment's focus on IT infrastructure and processes (architecture, processes, and skills) to alignment of the central operational subunits within the IT function: development and operations (Henderson & Venkatraman, 1993; Onita & Dhaliwal, 2011). Our model explains how organisations can align their IT functions through DevOps to meet rapidly changing requirements and fast-moving technological trends in a world of complex and intertwined IT architecture and processes (Krancher et al., 2018). In the following, we discuss the emergent alignment mechanisms and explain how they interrelate and resolve misalignment.

Our findings reveal that DevOps provides several mechanisms to align development and operations (Hemon et al., 2018; Maruping et al., 2009; Tiwana, 2018). We know from prior studies that alignment within the IT function will lead to better management of software engineering tasks (Dhaliwal et al., 2011). Extant literature notes that rapidly deploying new software functionality to customers weakens the software stability due to confusion among IT workers (Onita & Dhaliwal, 2011). Thus, confronted with the problem of achieving high software stability and innovation power, our findings suggest the antecedents of alignment in DevOps to accomplish team effectiveness and a high level of intra-IT alignment. In the following, we discuss how these three mechanisms support intra-IT alignment.

5.1.1. Individual componentization

Individual componentization is the mechanism creates a flexible IT architecture that supports and requires integrated responsibility and multidisciplinary knowledge to achieve intra-IT alignment. Individual componentization supports alignment between development and operations functions as it allows management of the IT function as one coherent software product. A flexible IT architecture enables developers to conduct silent releases. These releases ensure that new product versions are integrated in the software product without customer interference or downtimes. Further, convertible infrastructure fosters an individual componentization, ensuring stability of the IT infrastructure, when including new releases (Fitzgerald & Stol, 2017). Overall, containerisation aligns development and operation in the IT function by ensuring interoperability between new and existing software elements on the level of technological infrastructure. While prior literature focus on app and platform architecture from a technological perspective (Tiwana, 2018), this research introduces the new components containerisation and silent releases that support and enable interoperability within cross-functional teams for the whole software development lifecycle

Furthermore, integrated responsibility and a flexible architecture facilitates self-service for DevOps team members and supports the teams by allowing them to serve their own architectural needs. The team is now responsible for building and running the IT architecture for managing their IT service.

In order for DevOps team members to managing convertible software architecture, they need multidisciplinary knowledge acquired by developing and sharing skills and knowledge in this environment. Since the skill set of members is always limited to a certain degree, the DevOps team can decide which standards to implement in their environment and orient their skill sets to these standards.

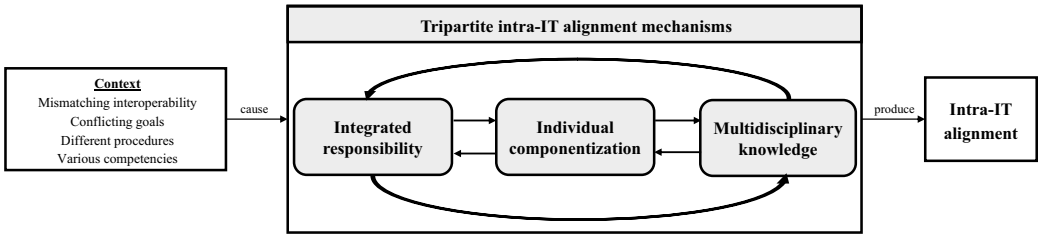

Figure 1. Tripartite intra-IT alignment.

5.1.2. Integrated responsibility

Integrated responsibility is the mechanism that creates accountability for the complete software delivery lifecycle within the IT function. It supports and requires individual componentization and multidisciplinary knowledge to achieve intra-IT alignment. Since the DevOps team is responsible for the software end-to-end, the team needs a high degree of freedom to resolve problems along the whole software lifecycle, spanning development and operations. Our concept broadens previous studies recommending collective ownership of software development and business processes through agile methodology by integrating the operations perspective (Maruping et al., 2009). Intra-IT alignment further extends the concept of software development dependency awareness (Bick et al., 2017). We show that successful DevOps approaches not only increase alignment between planning dependencies, but also have long-term operations consequences. We thereby add a long-term perspective to arguments provided in existing agile literature (Bick et al., 2017; Tiwana & Konsynski, 2010).

The integrated responsibility mechanism extends agile concepts beyond development into operations. First, DevOps teams schedule time on their backlog for unplanned operations work (Lee & Xia, 2010). Second, an individual componentized infrastructure enables technical compatibility, because high level of componentization increases IT alignment and enhances IT agility (Tiwana & Konsynski, 2010). Third, integrating multidisciplinary knowledge supports activities such as problem-solving and are no longer handled by separate subunits, but rather by the DevOps team. In summary, integrated responsibility resolves procedure misalignment by fostering common and decision-making processes in the team and thereby extends the boundaries of the agile method to software operations (Fitzgerald et al., 2006).

5.1.3. Multidisciplinary knowledge

The multidisciplinary knowledge mechanism creates a broad and adequate skillset and shared knowledge for conducting activities relevant to software development. It thereby supports individual componentization and individual responsibility to achieve intra-IT alignment. Acquiring multidisciplinary knowledge in different areas of expertise helps align development and operations as it integrates competencies within the DevOps team. For example, if a legacy system needs to be moved to modern cloud environment, setting up a team of people with different backgrounds will facilitate the move. Existing research highlights that autonomy and diversity is a key of teams (Kude et al., 2019; Lee & Xia, 2010). In DevOps teams is it essential that the team members have specialist knowledge in a certain area and that they broaden their knowledge to support and fill in for each other. Our results suggest that multidisciplinary knowledge facilitates a shared understanding of problems in the team and better enables team members to back up team members who have deeper knowledge in other areas.

Multidisciplinary knowledge enhances the development of knowledge and skills to manage individual componentization. Since the team is responsible for managing the complete software delivery lifecycle, mechanisms such as competency broadening foster developing team members by encouraging them to develop new capabilities e.g., in technology.

In addition, multidisciplinary knowledge facilitates integrated responsibility. The DevOps team retains ownership of software after it is deployed, it is motivated to solve problems proactively. Hence, innovation and stability are enabled through short decision-making processes within the team (Krancher et al., 2018).

Operational alignment research describes the linkage between business infrastructure and processes and IT infrastructure and processes (Gerow et al., 2014). This study applies the operational alignment perspective to IT functions and describes the linkages between the IT development and operations subunits and provides a new perspective on interoperability in operational alignment (Onita & Dhaliwal, 2011). Our model explains how cross-functional teams achieve common goals, how procedures are institutionalised and communication as well as knowledge gaps between development and operations subunits are filled (Bharadwaj et al., 2007; Vermerris et al., 2014; Wagner et al., 2014).

5.2. The importance of intra-IT alignment

Our research highlights the importance of operational alignment between IT development and operations and suggests different mechanisms to resolve misalignment within the IT function. The alignment mechanism individual componentization addresses intra-IT misalignment in interoperability (Mis-A4). The case of DevOps illustrates that both technological artefacts such as a continuous deployment pipeline but also system design elements such as APIs and a convertible architecture increase the operability of newly developed software components and the existing system landscape (Edberg et al., 2012; Onita & Dhaliwal, 2011). Individual componentization thereby reduces interoperability misalignment between development and operations IT subunits.

Our findings also indicate that the alignment mechanisms of individual componentization and integrated responsibility resolve goal misalignment (Mis-A1) in the IT function (Fichman & Melville, 2014). The DevOps example illustrates that the combination of accountability and autonomy will enable the joint DevOps team to develop shared goals that meet both

development and operational requirements. The seemingly conflicting goals of innovation and stability in development and operations IT subunits are integrated through the components containerisation and competency broadening by building common responsibility within the team. Our results thereby explain how goal-oriented operational alignment can be achieved (Martin et al., 2008).

In addition, this research adds value to the procedures perspective in operational alignment (Chan & Reich, 2007; Vermerris et al., 2014). This study highlights that the alignment mechanisms integrated responsibility and multidisciplinary knowledge address procedural misalignment (Mis-A2). First, we illustrate the value of ensuring procedural alignment across the IT function (Edberg et al., 2012). The implementation of DevOps teams exemplifies the creation of value in the daily business of development and operations procedures through extending agile, process automation, product orientation, and problem ownership. In terms of operational processes, the reactive orientation of IT operations is shifted to a more proactive orientation (Forsgren et al., 2018). This helps align underlying processes to achieve both greater flexibility (development) and greater stability (operations).

Finally, our results contribute to interoperability in operational alignment (Martin et al., 2008; Wagner et al., 2014). We illustrate that the alignment mechanisms of multidisciplinary knowledge addresses misalignment in competencies (Mis-A3). Our results show that misalignment of competencies is common (Edberg et al., 2012). We show that communication and knowledge sharing in DevOps teams are enabled by competency broadening and skill distribution, as all team members take responsibility for all end-to-end development and operations activities. This fosters shared understanding and leads to better performance (Kude et al., 2019). Hence, the different competencies of development and operations are aligned.

In summary, we offer three alignment mechanisms to explain how DevOps fosters intra-IT alignment. Our tripartite intra-IT alignment model extends operational IT alignment to the development and operations functions (Dhaliwal et al., 2011; Onita & Dhaliwal, 2011). It expands the traditional limits of the SAM by considering intra-IT operational alignment (Henderson & Venkatraman, 1993). As DevOps teams abandon project structures and iterative phases, they are appealing targets for research in response to recent calls to investigate alignment between autonomous teams, responsibility for on-demand task management, and organisational goals.

5.3. Implications for practice

Table 6 below provides practical guidelines for aligning development and operations in joint DevOps teams.

5.4. Limitations and recommendations for further research

As with all research, this study is limited in several ways, which has implications for future research. First, despite our best efforts choose a wide array of interview partners, further research should test the applicability of our findings in other contexts (Glaser, 1978). In addition, complementary research is needed to identify other non-operational alignment dimensions that influence IT alignment. Beyond our narrow focus on operational intra-IT alignment, our results also provide initial insights into the social dimension of alignment (Wagner et al., 2014), which needs further amplification. Our research examines internally organised DevOps teams that are accountable for smaller software products like online shops. More complex software products, sourcing options and inter-organisational relationships are worthy of future research. Our findings are based on empirical accounts of DevOps. We also see the potential benefit of an in-depth analysis of other approaches to integrate operational units into the IT function. Lastly, future research could build upon our findings using data generated by other qualitative research methods, such as observation validated with quantitative methods. For example, we recommend examining how alignment mechanisms change within inter-organisational relationships – e.g., outsourcing options – and how these are related the domains of SAM.

Table 6. Practical guidelines and recommendation.

Practical guidelines	Recommendations
Dismantle monolithically IT architecture landscape	We recommend setting up an IT architecture that supports a high level of self-management within the DevOps teams. A convertible IT architecture enables teams to managed their services and develop an IT infrastructure that best supports processes. Investments should both modernise the IT landscape and support self-organisation and management within the team.
Integrate cross-functional teams	We recommend integrating cross-functional teams with end-to-end responsibility for the delivery lifecycle of one or more IT products. Their activities should be product-oriented rather project-oriented to achieve a high degree of coherence and social responsibility within team.
Enable knowledge sharing and mutual learning	We recommend integrating standards for knowledge sharing and learning opportunities within the team so that team members can broaden their knowledge and support each other. This implies adapting the knowledge needed for plan, build, and run activities related and limited to the specific products.

6. Conclusion

Our study addresses an important issue for IT functions: *What are the mechanisms by which development and operations IT functions achieve intra-IT alignment?* This study identifies three mechanisms for resolving misalignment between development and operations in DevOps teams: individual componentization, integrated responsibility, and multidisciplinary knowledge. Each of these mechanisms contain several components that help resolve intra-IT misalignment. We demonstrate the concept of tripartite intra-IT alignment by providing alignment mechanisms within DevOps that are linked to the operational level.

Acknowledgment

We thank the special issue editors Sabine Matook, Likoebe Maruping, and Knut Rolland, for their constructive and developmental feedback and guidance. We thank also the associate editor and three anonymous reviewers who provided valuable insights that helped us improve and refine our paper. We are grateful to the practitioners who took time out of their busy diaries to inform our research. The research for this paper was financially supported by German Federal Ministry of Education and Research (BMBF) under grant code [03FH005PX4]. The full responsibility for the manuscript remains with the authors.

Disclosure statement

No potential conflict of interest was reported by the authors.

Notes on contributors

Anna Wiedemann is a Research Associate at Neu-Ulm University of Applied Sciences and a doctoral candidate at the Technische Universität München (TUM) in Germany. Her current research comprises agile IT, DevOps teams, and qualitative research methods. Anna Wiedemann has already published her research in the Communications of the ACM and a number of refereed conference proceedings, including but not limited to the International Conference on Information Systems (ICIS), the European Conference on Information Systems (ECIS), and the Hawaii International Conference on System Sciences (HICSS).

Manuel Wiesche is full professor and chair of Digital Transformation at TU Dortmund University. He graduated in Information Systems from Westfälische Wilhelms-Universität, Münster, Germany and holds a doctoral degree and a habilitation degree from TUM School of Management, Technische Universität München, Munich, Germany. His current research interests include IT workforce, IT project management, digital platform ecosystems, and IT service innovation. His research has been published in MISQ, JMAR, CACM, I&M, EM, and MISQE.

Heiko Gewald is Research Professor of Information Management at Neu-Ulm University of Applied Sciences in Germany. He received his PhD in Information Systems from Goethe University Frankfurt and holds a Master's degree in Business Administration from University of Bamberg and a European Master of Business Science from Heriot-Watt University, Edinburgh. His research focuses on the adoption and use of Information Systems in healthcare, Internet usage of the ageing society, outsourcing, IT strategy and project management. He frequently speaks at academic and practitioner-oriented conferences on these subjects. His work has been published in the European Journal of Information Systems, Health Systems, Information & Management, Journal of Economic Commerce Research, Information Systems Frontiers, Communications of the ACM and numerous other journals.

Helmut Krcmar is Professor of Information Systems, Department of Informatics, at Technische Universität München (TUM) with a joint appointment to the School of Management. His work experience includes a Post-Doctoral Fellowship, IBM Los Angeles Scientific Center, and Assistant Professor of Information Systems, Leonard Stern School of Business, NYU, and Baruch College, CUNY. From 1987 to 2002 he was Chair for Information Systems, Hohenheim University, Stuttgart, where he served as Dean, Faculty of Business, Economics and Social Sciences. Helmut's research interests include information and knowledge management, engineering, piloting, and management of innovative IT-based services, computer support for collaboration in distributed and mobile work and learning processes. Helmut co-authored a plethora of research papers published in major IS journals including MISQ, JMIS, JIT, JSIS, ISJ, I&M, CAIS, TOCHI and BISE. In Germany, his book "Information Management" is now in a 6th edition (2015). Interdisciplinary work incorporates areas such as accounting, mechanical engineering, and health care. Helmut collaborates in research with a wide range of leading global organizations. He is a Fellow of the Association of Information Systems (AIS) and member of acatech – National Academy of Science and Engineering

ORCID

Anna Wiedemann http://orcid.org/0000-0002-3535-5317
Manuel Wiesche http://orcid.org/0000-0003-0401-287X
Heiko Gewald http://orcid.org/0000-0003-2107-2217
Helmut Krcmar http://orcid.org/0000-0002-2754-8493

References

Barua, A., Konana, P., Whinston, A. B., & Yin, F. (2004). An empirical investigation of net-enabled business value. *MIS Quarterly*, *28*(4), 585–620. https://doi.org/10.2307/25148656

Bharadwaj, S., Bharadwaj, A., & Bendoly, E. (2007). The performance effects of complementarities between information systems, marketing, manufacturing, and supply chain processes. *Information Systems Research*, *18*(4), 437–453. https://doi.org/10.1287/isre.1070.0148

Bick, S., Spohrer, K., Hoda, R., Scheerer, A., & Heinzl, A. (2017). Coordination challenges in large-scale software development: A case study of planning misalignment in hybrid settings. *IEEE Transactions on Software Engineering*, *44*(10), 932–950. doi:10.1109/TSE.2017.2730870.

Birks, D. F., Fernandez, W., Levina, N., & Nasirin, S. (2013). Grounded theory method in information systems research: Its nature, diversity and opportunities. *European Journal of Information Systems*, *22*(1), 1–8. https://doi.org/10.1057/ejis.2012.48

Chan, Y. E., & Reich, B. H. (2007). IT alignment: what have we learned? *Journal of Information Technology*, *22*(4), 297–315. https://doi.org/10.1057/palgrave.jit.2000109

Constantinides, P., & Barrett, M. (2014). Information infrastructure development and governance as collective action. *Information Systems Research*, *26*(1), 40–56. https://doi.org/10.1287/isre.2014.0542

Cram, A., & Newell, S. (2016). Mindful revolution or mindless trend? Examining agile development as a management fashion. *European Journal of Information Systems*, *25*(2), 154–169. https://doi.org/10.1057/ejis.2015.13

Dhaliwal, J., Onita, C. G., Poston, R., & Zhang, X. P. (2011). Alignment within the software development unit: Assessing structural and relational dimensions between developers and testers. *The Journal of Strategic Information Systems*, *20*(4), 323–342. https://doi.org/10.1016/j.jsis.2011.03.001

Edberg, D., Ivanova, P., & Kuechler, W. (2012). Methodology mashups: An exploration of processes used to maintain software. *Journal of Management Information Systems*, *28*(4), 271–304. https://doi.org/10.2753/MIS0742-1222280410

Eisenhardt, K. M. (1989). Building theories from case study research. *Academy of Management Review*, *14*(4), 532–550. https://doi.org/10.5465/amr.1989.4308385

Fichman, R. G., & Melville, N. P. (2014). How posture-profile misalignment in IT innovation diminishes returns: conceptual development and empirical demonstration. *Journal of Management Information Systems*, *31*(1), 203–240. https://doi.org/10.2753/MIS0742-1222310109

Fitzgerald, B., Hartnett, G., & Conboy, K. (2006). Customising agile methods to software practices at intel shannon. *European Journal of Information Systems*, *15*(2), 200–213. https://doi.org/10.1057/palgrave.ejis.3000605

Fitzgerald, B., & Stol, K.-J. (2017). Continuous software engineering: A roadmap and Agenda. *Journal of Systems and Software*, *123*, 176–189. https://doi.org/10.1016/j.jss.2015.06.063

Forsgren, N., Humble, J., & Kim, G. (2018). *The science behind DevOps: Accelerate building and scaling high performing technology organizations*. IT Revolution.

Gerow, J. E., Grover, V., Thatcher, J. B., & Roth, P. L. (2014). Looking toward the future of IT-business strategic alignment through the past: A meta-analysis. *MIS Quarterly*, *38*(4), 1059–1085. https://doi.org/10.25300/MISQ/2014/38.4.10

Glaser, B. G. (1978). *Theoretical sensitivity: Advances in the methodology of grounded theory*. The Sociology Press.

Glaser, B. G., & Strauss, A. L. (1967). *The discovery of grounded theory: Strategies for qualitative research*. Aldine Publishing Company.

Gregory, R. W., Kaganer, E., Henfridsson, O., & Ruch, T. J. (2018). IT consumerization and the transformation of IT governance. *MIS Quarterly*, *42*(4), 1225–1253. doi: 10.25300/MISQ/2018/13703

Hemon, A., Monnier-Senicourt, L., & Rowe, F. (2018). *Job satisfaction factors and risks perception: An embedded case study of devops and agile teams*. *Paper presented at the International Conference on Information Systems*, San Francisco.

Henderson, J. C., & Venkatraman, N. (1993). Strategic Alignment: Leveraging information technology for transforming organizations. *IBM Systems Journal*, *32*(1), 4–16. https://doi.org/10.1147/sj.382.0472

Huber, T. L., Kude, T., & Dibbern, J. (2017). Governance practices in platform ecosystems: Navigating tensions between cocreated value and governance costs. *Information Systems Research*, *28*(3), 563–584. https://doi.org/10.1287/isre.2017.0701

Kang, S., Park, J.-H., & Yang, H.-D. (2008). ERP alignment for positive business performance: Evidence from Korea's ERP market. *Journal of Computer Information Systems*, *48*(4), 25–38. doi: 10.1080/08874417.2008.11646032

Kim, C., & Westin, S. (1988). Software maintainability: Perceptions of EDP professionals. *MIS Quarterly*, *12*(2), 167–185. https://doi.org/10.2307/248841

Krancher, O., Luther, P., & Jost, M. (2018). Key affordances of platform-as-a-service: Self-organization and continuous feedback. *Journal of Management Information Systems*, *35*(3), 776–812. https://doi.org/10.1080/07421222.2018.1481636

Kude, T., Mithas, S., Schmidt, C. T., & Heinzl, A. (2019). How pair programming influences team performance: The role of backup behavior, shared mental models, and task novelty. *Information Systems Research*, *30*(4), 1145–1163. https://doi.org/10.1287/isre.2019.0856

Lee, G., & Xia, W. (2010). Toward agile: An integrated analysis of quantitative and qualitative field data on software development agility. *MIS Quarterly*, *34*(1), 87–114. https://doi.org/10.2307/20721416

Markus, M. L., & Keil, M. (1994). If we build it, they will come: Designing information systems that people want to use. *Sloan Management Review*, *35*(4), 11–35. https://sloanreview.mit.edu/article/if-we-build-it-they-will-come-designing-information-systems-that-people-want-to-use/

Martin, S. F., Wagner, H.-T., & Beimborn, D. (2008). *Process documentation, operational alignment, and flexibility in IT outsourcing relationships: A knowledge-based perspective*. *Paper presented at the International Conference on Information Systems*, Paris, France.

Maruping, L., & Matook, S. (forthcoming). The multiplex nature of the customer representative role in agile information systems development. *MIS Quarterly*. https://misq.org/skin/frontend/default/misq/pdf/Abstracts/12284_RA_Maruping_Abstract.pdf

Maruping, L., Venkatesh, V., & Agarwal, R. (2009). A control theory perspective on agile methodology use and changing user requirements. *Information Systems Research*, *20*(3), 377–399. https://doi.org/10.1287/isre.1090.0238

Myers, M. D., & Newman, M. (2007). The qualitative interview in IS research: examining the craft. *Information and Organization*, *17*(1), 2–26. https://doi.org/10.1016/j.infoandorg.2006.11.001

Nadler, D., & Tushman, M. (1993). A general diagnostic model for organizational behavior: Applying a congruence perspective. In J. R. Hackman, E. E. Lawler, & L. W. Porter (Eds.), *Perspectives on behavior in organizations* (pp. 112–124). McGraw-Hill.

Onita, C., & Dhaliwal, J. (2011). Alignment within the corporate IT unit: An analysis of software testing and development. *European Journal of Information Systems*, *20*(1), 48–68. https://doi.org/10.1057/ejis.2010.52

Powell, T. C. (1992). Organizational alignment as competitive advantage. *Strategic Management Journal*, *13*(2), 119–134. https://doi.org/10.1002/smj.4250130204

Ramesh, B., Cao, L., & Baskerville, R. (2010). Agile requirements engineering practices and challenges: An empirical study. *Information Systems Journal*, *20*(5), 449–480. https://doi.org/10.1111/j.1365-2575.2007.00259.x

Reynolds, P., & Yetton, P. (2015). Aligning business and IT strategies in multi-business organizations. *Journal of Information Technology*, *30*(2), 101–118. https://doi.org/10.1057/jit.2015.1

Rivard, S., Raymond, L., & Verreault, D. (2006). Resource-based view and competitive strategy: An integrated model of the contribution of information technology to firm performance. *The Journal of Strategic Information Systems*, *15*(1), 29–50. https://doi.org/10.1016/j.jsis.2005.06.003

Sarker, S., & Sarker, S. (2009). Exploring agility in distributed information systems development teams: An interpretive study in an offshoring context. *Information Systems Research*, *20*(3), 440–461. https://doi.org/10.1287/isre.1090.0241

Shpilberg, D., Berez, S., Puryear, R., & Shah, S. (2007). Avoiding the alignment trap in IT. *MIT Sloan Management Review*, *49*(1), 51. https://sloanreview.mit.edu/article/avoiding-the-alignment-trap-in-it/

Swanson, B. E., & Beath, C. M. (1989). Reconstructing the systems development organization. *MIS Quarterly*, *13*(3), 293–307. https://doi.org/10.2307/249004

Tiwana, A. (2018). Platform synergy: Architectural origins and competitive consequences. *Information Systems Research*, *29*(4), 829–848. https://doi.org/10.1287/isre.2017.0739

Tiwana, A., & Konsynski, B. (2010). Complementarities between organizational IT architecture and governance structure. *Information Systems Research*, *21*(2), 288–304. https://doi.org/10.1287/isre.1080.0206

Urquhart, C. (2012). *Grounded theory for qualitative research: A practical guide*. SAGE Publication Inc.

Vermerris, A., Mocker, M., & van Heck, E. (2014). No time to waste: the role of timing and complementarity of alignment practices in creating business value in IT projects. *European Journal of Information Systems*, *23*(6), 629–654. https://doi.org/10.1057/ejis.2013.11

Wagner, H.-T., Beimborn, D., & Weitzel, T. (2014). How social capital among information technology and business units drives operational alignment and IT business value. *Journal of Management Information Systems*, *31*(1), 241–272. https://doi.org/10.2753/MIS0742-1222310110

Walsham, G. (1995). Interpretive case studies in IS research: Nature and method. *European Journal of Information Systems*, *4*(2), 74–81. https://doi.org/10.1057/ejis.1995.9

Wiedemann, A., Forsgren, N., Wiesche, M., Gewald, H., & Krcmar, H. (2019). Research for practice: The DevOps phenomenon. *Communications of the ACM*, *62*(8), 44–49. https://doi.org/10.1145/3331138

Wiedemann, A., Wiesche, M., Thatcher, J. B., & Gewald, H. (2019). *A control-alignment model for product orientation in DevOps teams–A multinational case study. Paper presented at the International Conference on Information Systems*, Munich, Germany.

Wiesche, M., Jurisch, M. C., Yetton, P. W., & Krcmar, H. (2017). Grounded theory methodology in information systems research. *MIS Quarterly*, *41*(3), 685–701. https://doi.org/10.25300/MISQ/2017/41.3.02

Yin, R. K. (2018). *case study research and applications: Design and methods* (Vol. 6). SAGE Publication Inc.

Appendix A. Description of firms in our study

Case	DevOps Team Setup
Case 1, a leading food and convenience retail company: The company is organised by stores and online shopping sales channels. They use DevOps to develop internal products and services, e.g., an app for managing a delivery service as well as other IT services.	• Mainly development background • Feature development, automation, monitoring tasks • 24/7 decentralised service support • Scrum principles for tasks organisation • Five team members, product owner, agile coach
Secondary data: Elaboration a team structure sketch, blog articles, conference presentations, and publications	
Case 2, a leading financial banking institution: The institution offers various types of banking products and online banking. They started integrating DevOps principles for a securities management system. The team has a group manager and fifteen team members.	• Mainly operations background • Test automation, release management, monitoring tasks • 24/7 service support • Agile-traditional hybrid approach for tasks organisation
Secondary data: Onsite-visit and observations, elaboration of sketch of the team structure, flipchart diagrams, product information, and publication	
Case 3, a leading insurance company: The company offers various types of insurance through different sales channels. Their DevOps team operates an internal delivery platform. Responsible persons are a team lead, product owner and eight team members	• Mainly development background • Features development, monitoring, security tasks • Infrastructure set-up • 24/7 service support • Scrum principles for tasks organisation
Secondary data: Onsite-visit and observations, elaboration of sketch of the team structure, conference presentation, and publications	
Case 4, an Internet company with more than 1,000 employees. Their services help end customers identify, compare, and buy products. The team consists of a team lead and eight team members. They manage their IT organisation with DevOps.	• Mainly development background • Features development, automation monitoring tasks • 24/7 service support and by team lead at night • Kanban principles for tasks organisation
Secondary data: Onsite-visit and observations, elaboration of sketch of the team structure, flipchart diagrams, YouTube videos, and publications	
Case 5, a leading retail company with more than 50,000 employees. The company uses different sales channels (e.g., shops and online). Internal DevOps team has seven team members, product owner, and a team lead teams that manage their online shop.	• Mainly development background • Features development, quality assurance, automation, monitoring tasks • Infrastructure management • 24/7 service support • Scrum principles for tasks organisation.
Secondary data: Onsite-visit and observations, elaboration of sketch of the team structure, flipchart diagrams, and publications	
Case 6, a warehousing business with more than 20,000 employees. The company has stores and online shops as sales channels. The DevOps team consists of a team lead with six team members that runs their online shop and manages the basis platform.	• Mainly operations background • Developing, platform support, test automation tasks • Infrastructure set-up, scripting, automation • 24/7 service support • Kanban principles for tasks organisation
Secondary data: Elaboration of sketch of the team structure, blog articles, and publications	
Case 7, a leading foods and convenience retailer with more than 100,000 employees. Its sales channels are stores and an online shop. The company started transforming some teams to DevOps, e.g., the configuration management tool	• Mainly operations background • Infrastructure set-up, scripting, automation • 24/7 service support • Hybrid approaches for task organisation • Team lead, three team members
Secondary data: Onsite-visit and observations, elaboration of sketch of the team structure, flipchart diagrams, and publications	
Case 8, a well-known online travel agency with more than 1,000 employees. Serves customers via an online shop. The DevOps team has one team lead and five members to organises their internal online store platform.	• Mainly operations background • Infrastructure set-up, automation, scripting, system configuration tasks • 24/7 service support • Scrum principles for tasks organisation
Secondary data: Elaboration of sketch of the team structure, company presentation, blog articles, and publications	

Appendix B

Interview Questions (Excerpt)

Personal and organisational demographics

Please introduce yourself (background, education, experience, role, etc.)?
How is your organisation structured (organigram, staff, management, etc.)?

Team-related issues

(a) Product and activities:

How do you structure your DevOps team? Please explain how the DevOps is set up and why?
Which product(s)/service(s) are you responsible for?
For which tasks and activities is the DevOps team responsible for?
What do you consider as essential components of a DevOps team?
What are the similarities and differences of managing IT development and IT operations activities in the team?

(a) Personal development and training:

How does the company train and develop the DevOps team members personally and professionally?
How are you staffing DevOps teams?
What skills and knowledge do you have to learn?
How is knowledge shared within the DevOps team and the company?
Which skills are you integrating in the DevOps teams?

(a) Architecture and methods:

Please describe the IT architecture for your product?
How are you maintaining your product?
Which tools are you using and why?
Are you using agile software development methods and why?
How are you integrating operations/development into your team?

(a) Others:

What mechanisms do you use to align development and operations?
What are major challenges in achieving alignment?