Technische Universität München

Fakultät für Informatik

Lehrstuhl für Wirtschaftsinformatik

Prof. Dr. Helmut Krcmar

The Integration of DevOps Teams in Established IT Organizations
Effective Methods and Empirical Insights

Anna Maria Patricia Wiedemann, Master of Science

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:	Prof. Dr. Florian Matthes
Prüfer der Dissertation:	1. Prof. Dr. Helmut Krcmar
	2. Prof. Dr. Heiko Gewald
	3. Prof. Dr. Nils Urbach

Die Dissertation wurde am 20.04.2020 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 29.09.2020 angenommen.

Acknowledgements

First of all, my sincere thanks go to my supervisors Prof. Dr. Helmut Krcmar and Prof. Dr. Heiko Gewald. My Ph.D. advisor Prof. Dr. Helmut Krcmar has fully integrated me into the academic environment of his Chair of Information Systems at the Technical University of Munich. Due to the valuable guidance he motivated and empowered the progress of my dissertation as well as constant improvement of my work. Prof. Dr. Heiko Gewald encouraged me to join his research team at the University of Applied Sciences Neu-Ulm. He introduced me into Information Systems research and gave me his support to unfold and conduct my research at the national and international level. This dissertation was enabled through their supervision and vital discussions about the development of my projects and forward-looking thoughts. I am grateful that I had the privilege of having such outstanding personalities accompanying me professionally and humanly.

I am very thankful to my friends and former colleagues Thomas Schulz, Robert Rockmann, Nina Polzer, Sina Zimmermann, Maximilian Haug, and Anna Gaab from the University of Applied Sciences Neu-Ulm. I enjoyed working and debating with them. They spent valuable support in stressful times. Furthermore, I would like to thank the Chair of Information Systems at the Technical University of Munich, my mentor and research group leader Prof. Dr. Manuel Wiesche; his granularity to detail, discipline, professionalism and patience led to my course of navigation to success. It was a precious experience to collaborate with the Chair, particularly with Maximilian Schreieck, Christoph Pflügler, Michael Lang, Jörg Weking, Kai Klinker and Barbara Prommegger who patiently answered my questions.

In addition, I would like to express my gratitude to my co-authors Prof. Dr. Jason Thatcher, Dr. Nicole Forsgren, and Dr. Andy Weeger for their contributions and discussions during the preparation of our publications. Prof. Dr. Jason Thatcher from the Temple University excellently advised my research which led to a steady improvement of this thesis. Dr. Nicole Forsgren, the founder of DevOps Research and Assessment, made significant contributions to our publications. Because of her commitment, I have been able to win many case study partners for my research. Dr. Andy Weeger, my former colleague in Neu-Ulm, supported me especially at the beginning of my scientific work as a mentor, and he kindly explained the processes of Information Systems research to me. It was a pleasure to work with him.

Moreover, I would like to thank Prof. Dr. Eileen Trauth, Emeritus Professor at the Pennsylvania State University, who has accompanied me over the years and has given me numerous contacts from research and practice. Prof. Dr. Fred Niederman from the Saint Louis University, Prof. Dr. Jaime Windeler and Prof. Dr. Binny Samuel from the University of Cincinnati, Prof. Dr. Jason Cohen from the University of the Witwatersrand, Prof. Dr. Andrew Georgiou and Dr. Maria Dahm from the Macquarie University, Prof. Dr. Tim Weitzel from the University of Bamberg, Dr. Marten Risius from the University of Queensland and Prof. Dr. Damien Joseph from the Nanyang Technological University gave me meaningful feedback and helped to identify potential case study partners for my global investigation.

This research was supported by funding from the German Federal Ministry of Education and Research and by NTT DATA, which I would like to honor and appreciate. My friends and NTT DATA colleagues Dr. Margarete Sackmann, Markus Schlayer, and Christoph Böhm always emboldened me.

My very special thanks are devoted to my dear family, my grandparents Anna und Johann Gaßner, my mother Adelheid Wiedemann, my cousin Angelina Ghalayini, my aunt Maria Wiedemann, and the family of my father Josef Wiedemann who continuously reinforced and motivated me. Thank you for always being there for me.

My deepest gratitude is dedicated to my aunt Martina Gaßner. She has been supporting me my entire life, believed in me, and spent every effort in advising me. Thank you for your tremendous help and teaching me to be nearly restless in analyzing and achieving true worth and a kind heart, researching for the best results in life, and that I become an enrichment for my fellow men.

Munich, November 2020 Anna Wiedemann

Kurzfassung

Problemdarstellung: Aufgrund des digitalen Wandels passen traditionelle Funktionen der Informationstechnologie (IT) ihre Organisationsstrukturen an, um schneller auf sich ändernde Chancen, Kundenansprüche und Herausforderungen reagieren zu können. Etablierte Firmen streben nach einer raschen Markteinführung neuer Softwarefunktionen und Skalierbarkeit von Softwareinnovationen. Um Synergieeffekte und Wertschöpfung zu erzielen, setzen viele IT-Funktionen mit Silo-IT-Abteilungen funktionsübergreifende DevOps-Teams ein. DevOps unterstützt die systematische Zusammenarbeit zwischen Softwareentwicklern und -betrieb, indem beide Funktionen im Team für die Erstellung von digitalen Lösungen integriert werden. Diese Dissertation befasst sich mit der Frage, wie Unternehmen die organisatorischen, kulturellen, technologischen und menschlichen Herausforderungen bei der Integration von DevOps durch Theorien im Bereich Skills, der Informationssystem (IS)-Kontrolle sowie des Business-IT-Alignment bewältigen können.

Forschungsdesign: Die zugrunde liegende Philosophie dieser Dissertation ist die interpretative Erkenntnistheorie. Bei der Interpretation sozialer Praktiken werden in dieser Dissertation qualitative Forschungsmethoden und unter anderem die *Grounded Theory* angewendet. Die Ergebnisse aus mehreren Fallstudien werden mit Hilfe bestehender Literatur induktiv verallgemeinert.

Hauptergebnisse: Diese Forschung bietet Einblicke in die Integration von DevOps und die damit verbundenen Fähigkeiten, die in Modelle des Organisationsaufbaus, des Personalmanagements und der Mitarbeiterbewertung einfließen können. Insbesondere wurde die Notwendigkeit erkannt, Kontrollmechanismen aus der Software-Entwicklung und dem Software-Betrieb zu kombinieren. Daraus wurde ein neues, bivariates Managementmodell mit Mensch-zu-Technologie- und Mensch-zu-Mensch-Kontrollbeziehungen für das Management funktionsübergreifender Teams entwickelt. Gemeinsames Domänenwissen ist ein Schlüsselfaktor für eine abgestimmte Kontrolle innerhalb von DevOps-Teams. Darüber hinaus unterstreicht diese Arbeit den Wertbeitrag der Abstimmung von Softwareentwicklung und -betrieb innerhalb gemeinsamer DevOps-Teams. Schließlich führen die Mechanismen der individuellen Komponentenbildung, der integrierten Verantwortung und des multidisziplinären Wissens zu einem intra-IT-Alignment-Modell.

Limitationen: In dieser Arbeit wurden hauptsächlich Personen aus der IT mit Hilfe von Experteninterviews befragt. In funktionsübergreifenden Teams sind zuweilen auch Personen aus Fachbereichen integriert. Daher sollte in Zukunft die Integration der Fachbereiche in Betracht gezogen werden.

Weitere Forschung: Diese Dissertation zeigt verschiedene Möglichkeiten für weitere Forschung auf. Die dargestellten Kontroll- und Alignment-Modelle können durch quantitative Studien validiert werden. In dieser Forschung wurde gezeigt, dass DevOps ein globales Konzept ist, das auf transnationale und interkulturelle Teams anwendbar ist. Durch zukünftige Untersuchungen können weitere Einblicke in die Herausforderungen und Möglichkeiten des DevOps Konzepts gewonnen werden, welche sich beispielsweise aus kulturellen und transnationalen Unterschieden der Teams ergeben.

Beitrag: Diese Dissertation befasst sich mit DevOps-bezogenen IS-Skills, der IS-Kontrolltheorie und der Business-IT-Alignment Theorie. Sie liefert empirische Belege und eine theoretische Grundlage, die Unternehmen zur Entwicklung erfolgreicher, in die IT-Funktion integrierte DevOps-Teams nutzen können. Die neuen theoretischen Erkenntnisse, die in dieser Dissertation vermittelt werden, ebnen den Weg für potenzielle weitere Forschung. Ferner liefert diese Dissertation einen umfassenden Überblick über die Bewältigung organisatorischer, kultureller, technologischer und menschlicher Herausforderungen durch die Integration von DevOps-Teams in bestehende IT-Funktionen.

Abstract

Problem Statement: In response to digital transformation, traditional information technology (IT) functions are pressured to adapt their organizational structures in order to react more quickly to changing opportunities, customer demands, and challenges. Incumbent firms strive for faster time-to-market delivery of new software features and scalability of software innovations. In order to achieve synergy effects and value creation, many traditional IT functions with silo IT subunits are implementing cross-functional DevOps teams. DevOps supports systematic collaboration between software developers and operations, integrating development and delivery of software features for digital solutions. This dissertation addresses how companies can overcome the organizational, cultural, technological, and human challenges during DevOps integration through the lens of IS skills, IS control, as well as IT alignment theory.

Research Design: The underlying philosophy of this dissertation is interpretive epistemology. In interpreting social practices, the dissertation applies qualitative research methods and, among others, conceptualist-grounded theory and techniques. Finally, the findings from multiple case studies are generalized inductively in alignment with extant theory.

Main Results: This research provides insights into DevOps setup and related capabilities that can inform models of organizational setup, human resource management, and employee evaluation programs. Specifically, there is a need to combine control mechanisms from software development and software operations. Therefore, this study introduces a new bivariate management model of human-to-technology and human-to-human control relationships to manage cross-functional teams. Shared domain knowledge is a key enabler of aligned control within DevOps teams. Furthermore, this thesis underscores the value of aligning software development and software operations within joint DevOps teams. Finally, mechanisms of individual componentization, integrated responsibility, and multidisciplinary knowledge lead to an intra-IT alignment model.

Limitations: In this thesis, mainly IT people participated in the case study research. Sometimes, cross-functional teams include business people as well. Hence, integrating the business view should be taken into account in the future. Furthermore, the findings of this study are based on qualitative data. In this regard, further validity could be achieved by verifying the depicted results with quantitative and other research methods. This research investigates internal DevOps teams of incumbent firms.

Further Research: This dissertation points to various avenues for further research. First, the control and alignment models could be validated by longitudinal and quantitative studies. Second, as a global concept applicable to transnational and cross-cultural teams, DevOps is well suited for future research into the challenges and opportunities presented by cultural and transnational differences.

Contributions: This dissertation addresses DevOps-related IS skills and capabilities, IS control theory, and business-IT alignment theory, providing empirical evidence and a theoretical foundation that organizations can use to develop successful DevOps teams integrated within the IT function. The new theoretical insights provided in this dissertation pave the way for potential avenues for further research. This investigation contributes by delivering a

comprehensive overview of managing organizational, cultural, technological, and human challenges during digital transformation by integrating DevOps teams within existing IS functions.

Table of Content

List of Abbreviations

ASD	Agile Software Development
Biz	Business
BizDevOps	Business Development Operations
CASE	Computer-aided software engineering
CIO	Chief Information Officer
CON	Conference
CTO	Chief Technology Officer
DevOps	Development and Operations
GTM	Grounded Theory Model
HOUSE	Holistic Overview on an Unique Set of Examinations
IC	Intellectual Capital
IS	Information Systems
IT	Information Technology
ITIL	Information Technology Infrastructure Library
JNL	Journal
No.	Number
P	Publication
RQ	Research Question
SAM	Strategic Alignment Model
SDLC	Service Delivery Lifecycle
SLA	Service Level Agreement
VHB	Verband der Hochschullehrer für Betriebswirtschaft e.V.

List of Figures

List of Tables

PART A

1 Introduction

1.1 Motivation

Traditional information systems (IS) functions typically have separate functional subunits (Hemon, Monnier-Senicourt, & Rowe, 2018). As IS functions have grown exponentially over the past years, there is the need for closer cooperation between the information technology (IT) subunits to achieve agility and alignment throughout the complete software delivery lifecycle (Krancher, Luther, & Jost, 2018). Traditionally, the subunits of development and operations are organizationally divided within an IT function resulting in separated "silos". Subunits of software development and software operations sometimes have different and even opposing goals. For example, while software development strives to deliver new software functionalities and innovations quickly and efficiently, software operations prioritizes stability and continuity of running software applications as well as IT architecture (Edberg, Ivanova, & Kuechler, 2012; Krancher et al., 2018). These different goals can lead to misalignment between the two subunits. Despite these different goals, there are dependencies between software development and operations in various stages of the software delivery lifecycle (Fitzgerald & Stol, 2017; Slaughter, Levine, Ramesh, Pries-Heje, & Baskerville, 2006). Sharing capabilities, resources, and knowledge within internal IT subunits supports communication and facilitates the high performance of the IT function (Onita & Dhaliwal, 2011).

The pressure of rapid digital transformation has forced IT organizations to rethink their organizational structure so they can react quickly to changing opportunities, customer demands, and challenges (Sebastian et al., 2017). The changes are continual in IT and organizations, hence the alignment between business and IT must be enduring renewed and adjusted (Sabherwal & Chan, 2001). For example, IT functions are often working closely with business functions (Guillemette & Pare, 2012) to ensure permanently close alignment of the IT function with business goals (Sabherwal, Hirschheim, & Goles, 2001). To achieve this transformation, an increasing number of organizations have implemented the DevOps (Development and Operations) concept as a means to enhance collaboration between separated IT subunits through integration of cross-functional teams.

In 2009, the Belgian consultant Patrick Debois organized the first "DevOpsDays" conference. Since then, the term DevOps has been established (Debois, 2011). DevOps is a concept of systematic collaboration between software developers and software operations personnel to achieve faster deployments and operations of new software features for digital products and solutions (Hemon et al., 2018). DevOps provides guidelines for enhancing cooperation and achieving a higher level of stability (Forsgren, 2018; Hemon et al., 2018). The annual State of DevOps Report depicts that DevOps teams enable companies like CapitalOne to deploy software updates 50 times per day, and global innovators like Amazon and Netflix have hundreds of DevOps-oriented services which deploy up to 1,000 changes per day (Forsgren, Humble, Kim, Brown, & Kersten, 2017).

DevOps builds upon the agile software development journey that started in 2001 with the agile manifesto (Beck et al., 2001). DevOps uses the advantages of agile software developments to provide new code to customers quickly and to work in small teams. Including software operations tasks extends agile to software architecture and operations activities like release management (Hemon et al., 2018; Kim, Humble, Debois, & Willis, 2016). To achieve the speed and agility needed for innovation and transformation, 65 percent of CIOs surveyed in a recent study plan to expand agile and DevOps methods to large parts of the company by 2021 (Thorenz 2019).

Until now only few studies examine cross-functional teams within the IT function itself (Onita & Dhaliwal, 2011). The main target is to reach synergy effects by examining subunits of IT (e.g., software development, network operations, and architecture planning), to achieve a collaborative and integrative form to operate and communicate congruently within the IT function (Dhaliwal, Onita, Poston, & Zhang, 2011; Onita & Dhaliwal, 2011).

For this thesis, DevOps is defined as a cultural and technological concept for integrating the tasks, knowledge, and skills needed to plan, built and operate one or more IT service products of a cross-functional team. To ensure rapid delivery of new software features and innovations, to handle problems, and integrate operations activities, IT functions apply DevOps concepts to coordinate previously separated activities and to modernize IT architectures (Fitzgerald & Stol, 2017; Forsgren & Kersten, 2018).

1.2 Problem Statement

Despite multiple industry reports about the advantages of DevOps (e.g., Advance IT Minnesota (2016); Forsgren, Humble, Kim, Brown, and Kersten (2018a)), there has been relatively little research conducted that presents insights into the DevOps phenomenon (e.g., Hemon et al. (2018); Krancher et al. (2018)). By adopting a DevOps model, incumbent firms like Kaiser Permanente have significantly shortened their new software feature deployment cycles and become leaders in software innovation (Ross et al., 2016). Despite the proven potential of implementing the DevOps concept, doing so also challenges traditional IT organizations.

Forsgren, Humble, and Kim (2018) call for further investigation into how traditional organizations can resolve these challenges and achieve strategic advantage through managing IT services by DevOps. In nine papers, this cumulative dissertation applies various theories and research methods to analyze DevOps implementation in incumbent firms and presents recommendations for solving the organizational, technological, cultural and human challenges that typically arise during the organizational change.

Organizational Challenges:

As mentioned above, many established IT functions separate functional subunits (Hemon et al., 2018). One aim of IT functions is to implement cross-functional teams to enhance collaboration between different subgroups (Gregory, Kaganer, Henfridsson, & Ruch, 2018; Hemon et al., 2018). Though, reorganizing an IT function and integrating cross-functional teams responsible for end-to-end IT service provision can be challenging. The literature highlights that a service-centric operation model is suitable for designing IT functions (Sebastian et al., 2017). Companies can apply this model to restructure their IT functions, by integrating digital solutions for reducing discontinuities between software development and delivery (Fitzgerald & Stol, 2017; Sebastian et al., 2017).

Adopting cross-functional teams effects the strategic orientation of IT functions, increasing decentralization and requiring the adaption of formal processes and relational mechanisms. Fostering communication, shared understanding, and coordination among the stakeholders of cross-functional teams enable alignment between organizational strategy and objectives (Gregory et al., 2018). IS scholars suggest that the transformation of organizations can be a trigger for changes in the IS function (Guillemette & Pare, 2012). Although existing research presents insights into restructuring organizations during digital transformation (Jöhnk, Röglinger, Thimmel, & Urbach, 2017), further research is necessary to examine the role of DevOps integration in that transformation (Hemon et al., 2018).

Technology Challenges:

Incorporating DevOps teams into large and incumbent companies commonly also requires the adoption of new technologies, such as automation tools. Prior agile research finds that automation tools are often applied in software testing (Mangalaraj, Mahapatra, & Nerur, 2009; Onita & Dhaliwal, 2011). DevOps expands this focus to include automated build, deploy, release, and operating tasks (Fitzgerald & Stol, 2017). Fostering awareness of automation among all stakeholders is a key challenge because stakeholders are used to working with manual steps, e.g., acceptance testing. In addition, software development projects are seldom greenfield, but rather embedded in often large legacy systems with older software code. Integration with legacy code makes automation of processes, e.g., testing extremely difficult (Mangalaraj et al., 2009). Furthermore, implementing and managing new technologies and tools needed to achieve automation may require significant additional skills and resources (Fitzgerald & Stol, 2017; Stein, Galliers, & Markus, 2013).

Prior research concentrates on innovative IT architecture with a high level of modularity (Tiwana, 2018), but little is known about how small cross-functional teams can manage these modular architectures. Traditional IT functions manage their monolithic legacy systems with the help of data centers, and access is not easily given to developers (Vinekar, Slinkman, & Nerur, 2006). Today, agility is also applied to infrastructure, for example by authorizing stakeholders to manage their modular architecture environment through self-services (Yoo, Boland Jr, Lyytinen, & Majchrzak, 2012). Building on initial investigations into modular architectures, for example, in IS platform literature (e.g., Bush, Tiwana, & Rai, 2010; Tiwana, 2018), further investigation into the handling of modular architecture with a cross-functional team is needed.

Cultural Challenges:

Incumbent firms face cultural challenges arising during the integration of cross-functional teams. Managers have to bring team members closer together to realize team success (Chua, Lim, Soh, & Sia, 2012; Majchrzak, Malhotra, & John, 2005). The challenge is to create a culture of collaboration in a DevOps team among people with different backgrounds and perspectives. Prior research shows that social capital helps to enhance teamwork (Liu, Wang, & Chua, 2015). Barriers between software development and operations within cross-functional teams must be overcome to achieve effective collaboration and undertake complex tasks (Fitzgerald & Stol, 2017). This can be achieved by building strong partnership, shared values and understanding, and social cohesion (Liu et al., 2015).

Cross-functional teams are most efficient if they have a high level of autonomy to make their own decisions, such as about which technology best suits their needs (McAvoy & Butler, 2009). Even though most of the decisions are made within the group, in certain cases, decisions have to be made ad hoc by one person, e.g., when incidents happen at night. These decisions can impact the rest of the team. Hence, decentralized decision-making processes support team level choices and centralized hierarchical levels are needed for decisions with strategic impacts and dependencies on the other teams (e.g., budget relevance or technology shifts) (Heumann, Wiener, Remus, & Mähring, 2015). Building on initial investigations of social challenges in IS research in the area of project teams (Liu et al., 2015), further research into resolving the cultural challenges experienced by DevOps teams is required.

Human Challenges:

Extant literature stresses that the integration of cross-functional teams involves combining different competencies (Edberg et al., 2012) and requires extensive effort by IT leaders to build and create new values by linking existing capabilities with new ones (Ross et al., 2016). For many organizations, it is not clear which skills support the different tasks expected from a DevOps team. IT organizations are confronted with the challenge of having a broad skill set in a small cross-functional IT team instead of across several IT subunits. Team members need expertise in their specialist area but also strong generalist knowledge to successfully take on new tasks and develop high-quality software (Spohrer, Gholami, & Heinzl, 2012).

Within cross-functional teams, members are responsible for operations and support activities related to managing software solutions (Nelson, Nadkarni, Narayanan, & Ghods, 2000). When problems or failures appear outside of the teams (e.g., network errors) which cannot be traced back to a particular group or initiator, cross-functional teams must rely on other stakeholders to spot them and inform the corresponding person (Lempinen & Rajala, 2014). DevOps implementation confronts teams with knowledge aggregation issues, new tasks, and adoption of responsibility for failures, all of which demand research attention.

In summary, organizational, technological, cultural, and human challenges closely associated with DevOps implementation were analyzed. This cumulative dissertation develops recommendations for solving to these challenges.

1.3 Research Questions

The overall aim of this thesis is developing a theoretical and empirical understanding of how organizations can react to, manage, and govern challenges associated with DevOps implementation. The integration of cross-functional teams poses challenges for traditional-oriented IT functions, as outlined in section 1.2 above. Extensive efforts are necessary to build and create new value by linking existing capabilities with novel capabilities (Ross et al., 2016). Nevertheless, for many organizations, it is not clear which knowledge, capabilities competencies, and skills are needed to fulfil the different tasks expected from a DevOps team. Building up on existing research in the area of competency models (Matook & Maruping, 2014), this thesis provides a greater understanding of the key capabilities, knowledge, skills, and planning activities required in DevOps teams, as expressed by the following RQ:

RQ1: How can IT teams develop capabilities to respond efficiently to environmental changes and challenges?

In implementing DevOps teams, traditional IT functions in incumbent firms have to rethink their governance and control mechanisms to combine both development and operations tasks in team structures. Effective control is a critical factor for getting maximum business value from IT investments (Tiwana, 2009; Weill & Ross, 2004). Development control and operations control have different aims and profoundly different control philosophies (Edberg et al., 2012). The software development team's success is predominantly measured using time- and budget-related factors (Choudhury & Sabherwal, 2003; Cram & Newell, 2016), whereas research into IT operations management focuses on best practice approaches, e.g., the Information Technology Infrastructure Library (ITIL), a registered trademark. These guidelines focus primarily on internal operations goals, such as cost savings and process stability (Jia & Reich, 2013; Trusson, Doherty, & Hislop, 2014). In general, IT operations aim to avoid risks in running systems, but implementing changes through new software developments lead to possible threats (Arvidsson, Holmström, & Lyytinen, 2014).

Existing research has not comprehensively addressed the impact of initiated and executed governance and control arrangements (Sambamurthy & Zmud, 1999; Weill & Ross, 2004; Wiener, Mähring, Remus, & Saunders, 2016) for managing DevOps teams. Implementing DevOps is associated with control adaptation and how managers can interact with the DevOps team. This dissertation explains the transition from classical software development and operations control to DevOps control. The theoretical constructs presented in this thesis explain how cross-functional DevOps setting explains the resolutions of challenges through control mechanisms and how control can be implemented in DevOps teams. Therefore, RQ2 guided this thesis as follows:

RQ2: How can control mechanisms guide activities of software development and operations in DevOps teams?

IS research has concentrated extensively on strategic alignment (Gerow, Thatcher, & Grover, 2014b; Henderson & Venkatraman, 1993). However, less research has focused on operational alignment within the IT function (Dhaliwal et al., 2011; Slaughter et al., 2006). Onita and Dhaliwal (2011) find that alignment within the IT function supports communication and that a cooperative form of collaboration can facilitate congruent operations. Dhaliwal et al. (2011) point out the need to align internal IT subunits, which typically pursue different goals and are often misaligned.

IT functions face shorter development cycles, increasing demands from customers, and complex IT architectures (Fitzgerald & Stol, 2017; Slaughter et al., 2006). In response to these demands, many IT functions are implementing DevOps, taking a systematic approach to foster collaboration between IT units (Hemon et al., 2018). DevOps integrates the advantages of agile software development to react quickly to customer demands but also extends agility concepts to software architecture like infrastructure and systems (Krancher et al., 2018).

Cross-functional teams use methods for enhancing the collaboration between developers and users by building relationships between software developers from IT and product owners from business (Hemon et al., 2018; Matook, Soltani, & Maruping, 2016). Without including software operations into an agile environment, the classic paradigm of separated IT units will remain (Hemon et al., 2018). Accordingly, this research seeks to answer the following research question:

RQ3: How can development and operations activities be aligned in DevOps teams within the IT function?

1.4 Structure of the Thesis

This dissertation comprises nine papers, an introduction and a conclusion. The introduction provides a general overview of the thesis, presents the conceptional background and the theoretical perspectives as well as introduces the research methodologies. The conclusion summarizes the findings of the nine papers, discusses their contributions and implications for research and practice as well as identifies further research opportunities.

Furthermore, this dissertation conceptualizes cross-functional IT teams by investigating the setup, skills, and impacts of implementation of DevOps teams, exploring and analyzing these processes and proposing solutions to the challenges associated with them. The dissertation concentrates on how human, cultural, technological, and organizational aspects of DevOps manage challenges during and after implementation of the concept.

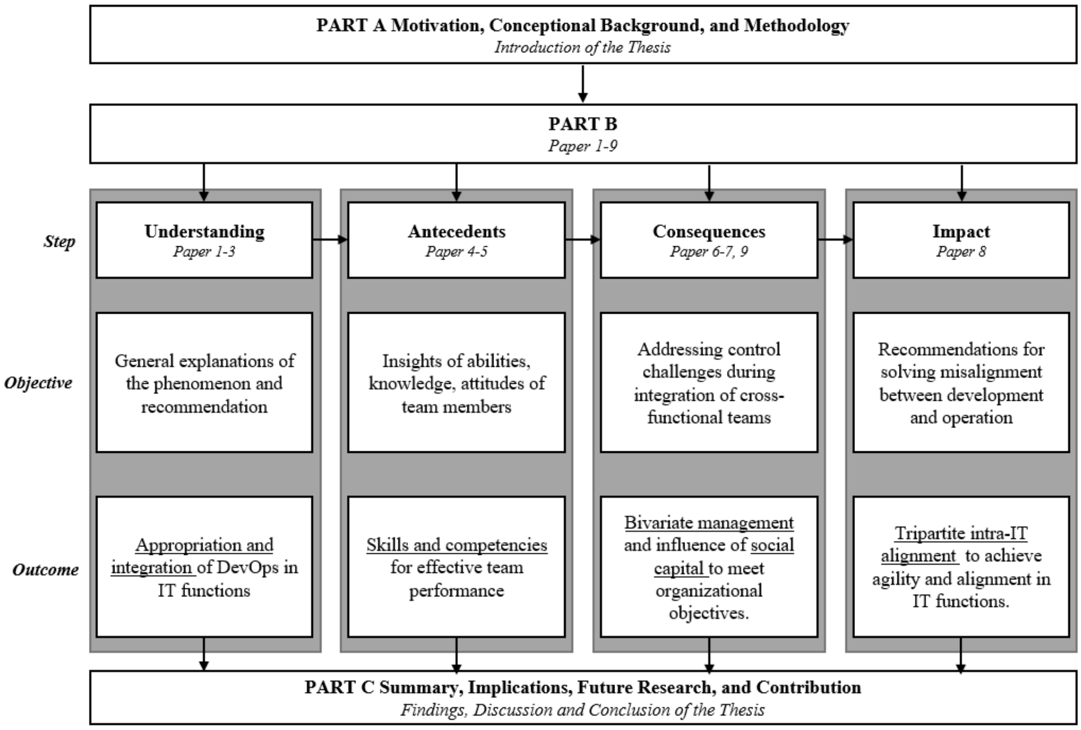

PART A Motivation, Conceptional Background, and Methodology
Introduction of the Thesis
PART B
Paper 1-9
Step
Objective
Outcome
Understanding
Paper 1-3
Antecedents
Paper 4-5
Consequences
Paper 6-7, 9
Impact
Paper 8
General explanations of the phenomenon and recommendation
Insights of abilities, knowledge, attitudes of team members
Addressing control challenges during integration of cross-functional teams
Recommendations for solving misalignment between development and operation
Appropriation and integration of DevOps in IT functions
Skills and competencies for effective team performance
Bivariate management and influence of social capital to meet organizational objectives.
Tripartite intra-IT alignment to achieve agility and alignment in IT functions.
PART C Summary, Implications, Future Research, and Contribution
Findings, Discussion and Conclusion of the Thesis

Figure 1. Summary of the Thesis (Own Depiction)

Figure 1 presents the structure of this thesis. Part A of this thesis gives an introduction about the overall research, the related literature and background, and the used research methodologies. The first step presents a general understanding of cross-functional IT teams (Paper 1-3). The second step depicts the antecedents for integrating cross-functional teams' key characteristics like skills and the integration of the planning process (Paper 4-5). The third step shows how control mechanisms need to be adapted as a result of reorganizing IT functions (Paper 6-7, 9). Part B1 includes published papers and part B2 includes papers under review. The fourth step visualizes the achievement of operational alignment within cross-functional teams to achieve agility throughout the complete IT function (Paper 8). Part C summarizes the overall findings of this thesis, with an overview of the main implications for research and practice. Finally, this thesis concludes with limitations and identifies opportunities for further research.

2. Conceptual Background

2.1 The Changing Role of IT Functions

Environmental changes have confronted IT functions for decades (Guillemette & Pare, 2012). Digital transformation challenges existing IT functions through new demands and technology trends. The role of the IT user is also altering. Originally, the IT function served primarily internal customers. Today, new stakeholders like external consumers have requirements that need to be considered (Leidner, Urbach, Drews, & Ross, 2017). New technologies have arisen that require continuous changes and transformation in IT and organizational environments for adjustment and alignment of business and IT objectives (Sabherwal et al., 2001).

Incumbent organizations manage their IT functions with the help of organizational silo units. People working in these silos are highly specialized in one area, and communication with other groups is rare. Managerial hierarchies, job roles, and work environments are clearly defined. Software updates are regularly planned, constructed, and disseminated under the leadership of command-and-control projects (e.g., waterfall approaches) (Dery, Sebastian, & van der Meulen, 2017; Vinekar et al., 2006). The provision of new software functionalities happens rarely, and large-scale releases are often planned for the weekend because of the need for extensive system outages during these releases (Forsgren & Humble, 2016; Vinekar et al., 2006).

Initially, traditional IT functions concentrate on the efficiency and provision of reliable scalable and secure IT services. Though IT is pervasive within the organization, it makes it challenging to provide reliable services to customers due to a rising number of users, devices, and methods. Traditional IT function set ups and processes require adaptions to meet the current business needs. There is a demand for optimization of simultaneous juxtaposition of digital innovations and delivering reliable and secure IT services (Haffke, Kalgovas, & Benlian, 2017).

Due to the challenges of digital transformation, IT functions are changing their mode of working within the internal organization and enable new forms of alignment and collaboration with the business units. The integration of cross-functional teams supports altering traditional IT organizations that were constituted as services providers and are now developing toward consultants, innovators, and enabler roles. The roles and elements of an IT function will be reflected in new processes, structures, methods, and governance mechanisms (Leidner et al., 2017).

2.1.1 Integrating DevOps Teams in IT Functions

The presentation of agile manifesto created a new awareness of software planning and development to provide better, faster, and secure IT services (Beck et al., 2001; Hemon et al., 2018; Humble & Farley, 2011). The agile movement impacted software development, and an increasing number of companies work successfully with agile principles. Unlike other parts of the IT functions, IT operations were less affected by agile methods. Since the introduction of DevOps, this concept provides new opportunities to broadening agility to other parts of the IT function. DevOps goes beyond agile software development and includes the operations part as well (Lwakatare et al., 2016).

Operations services are the provision of hardware, software, and supporting different IT services. ITIL presents guidelines for structuring operations processes. These guidelines are prevalent and widely spread in traditional firms (Bass, Weber, & Liming, 2015). ITIL supports the overall strategy, design, transition, operation, and continuous improvement in IT service concepts (Trusson et al., 2014). DevOps aims to reduce the commitment time between developers and operations people for providing new software features into the production area. Also, the rapid processing of service errors and the proactive monitoring and improvement of IT services is a major objective. In sum, the DevOps concept can use various ITIL processes in the form of continuous delivery rather than working with major releases for new software components (Bass et al., 2015).

DevOps bridges functional silo subunits of software development and operations by integrating them into one cross-functional team. These cross-functional teams are responsible for the value chain of an IT service, which includes planning, building, and running the software delivery lifecycles of digital solutions. Integrating cross-functional teams enable faster time-to-market of new software features through continuous delivery technology and higher quality of IT services as well as stability through integrating the operations area into the planning phase (Hemon et al., 2018; Krancher et al., 2018).

The literature highlights that a tighter collaboration between the development and the operations units of IT functions is required to ensure quick fixes of errors and to enhance the quality and reliability of running software (Hemon & Rowe, 2019). DevOps can help to combine these activities and reduce software deployment cycles by pushing new codes rapidly into the production environment (Fitzgerald & Stol, 2017). DevOps aims to generate synergy effects between the different IT units to avoid interruption of the software delivery process. Furthermore, DevOps enhances collaboration, virtualization, automation of tasks, and integration of tools for enabling a cross-functional work environment between development and operations (Fitzgerald & Stol, 2014; Reed, 2014).

2.1.2 The Role of Agility in IT Functions

DevOps combines the advantages of agile software development with software architecture and operations processes to react quickly to customer demands and broadens agility to the rest of the IT functions. Traditionally, agile software development methods accelerate the collaboration between developers and users by building relationships between developers from IT and product owners from business (Hemon et al., 2018; Matook et al., 2016). Including software operations in the agile environment, the traditional paradigm of separated IT units will be resolved (Hemon et al., 2018).

Compared to traditional plan-driven project management approaches, agile approaches offer a flexible alternative. The objectives of integrating agility within the IT are to respond more quickly and efficiently to customer demands and make work more transparent. Small projects are set up, and short iterations help the customer to understand the progress (Vinekar et al., 2006). Integrating client-driven priorities and constructive feedback cycles are valuable advantages of agile. More benefits are a faster turnaround, increased productivity, and higher software development satisfaction (Conboy, 2011; Vinekar et al., 2006).

Examinations on development (plan and build) show speed and creativity through projects (Maruping, Venkatesh, & Agarwal, 2009), while research on operations (run) consider the logics of stability, quality, and traceability of an architectural view (Krancher et al., 2018). DevOps fosters and combines agility between two different subunits by including the operations tasks and logic in one cross-functional team (Hemon et al., 2018; Krancher et al., 2018).

2.1.3 Summary of the Changing Role of IT

Managing digital transformation is challenging for incumbent companies. There is a need to evaluate the most beneficial approach for managing a software product before starting development and delivery. Prior research depicts that cross-functional approaches like DevOps are becoming more and more necessary to gain value in a competitive environment (Leidner et al., 2017; Ross et al., 2016). The recommendation for facing these challenges is to integrate new forms of collaborations such as cross-functional teams for business-critical IT services (Sebastian et al., 2017).

2.2 Theoretical Perspectives

This thesis analyzes the DevOps concept from various angles and theories which build upon each other. The following sections provides an overview of the theories applied in the papers included in this thesis.

2.2.1 Organizational Learning

Extant research defines organizational learning as the required new knowledge for the process of adoption and integration for realizing changes (Kang & Snell, 2009). Organizational learning includes gaining, sharing, and integrating knowledge to improve individual or group performance. Furthermore, organizational learning can take place on the individual, group, and organizational level. For integrating cross-functional teams like a DevOps team, organizational learning is a suitable approach to examine the ability and manifestation of ambidextrous learning within IT teams.

There are several investigations of organizational learning, concentrating on the approaches of exploration and exploitation (Kang, Morris, & Snell, 2007; Kang & Snell, 2009). Exploration is defined as the learning of new, external knowledge which is outside of company's knowledge domain, whereas exploitation focuses on improving and deepening existing internal company knowledge domains (March, 1991). Organizational learning is contingent on organizational intellectual capital (IC), which has human, social, and organizational dimensions (Kang & Snell, 2009). Cross-functional teams need both exploration as well as exploitation of knowledge to achieve a broad level of knowledge that fosters high performance. Based on existing literature a proposition for this thesis is that there are IC configurations that contribute to the ability of IT teams to develop both forms of organizational learning, exploration and exploitation. Thus, IC enable agility within the team. Table 1 presents an overview of the different dimensions of IC.

Intellectual capital dimension	Description	References
Human capital	Is defined as the combined knowledge of the individuals within a company. This dimension encompasses the knowledge, capabilities, competencies, and abilities which belong to and are used by the individuals. Human capital has a specialist or generalist manifestations.	Kang and Snell (2009); Subramaniam and Youndt (2005); Taylor and Greve (2006)
Social capital	Is defined as the knowledge available via relational networks. Social capital appears through the knowledge exchange within the firm. Firms gain advantages through sharing knowledge and insights within working groups and social relationships within or outside the company.	Kang and Snell (2009); Karahanna and Preston (2013); Nahapiet and Ghoshal (1998)
Organizational capital	Is defined as the institutional knowledge that is embedded in processes, structures, and systems. Organizational capital captures and integrates individual knowledge into organizational knowledge to be less dependent on individual people.	Grant (1996); Kang and Snell (2009)

Table 1. Overview of Intellectual Capital Dimensions (Own Depiction)

Organizational learning seems to be contingent on the IC of an organization or an entity like an IT team, respectively. Hence, in sum, there are IC configurations that facilitate organizational learning and help IT teams being agile.

2.2.2 IS Capabilities and the Concepts of Skills

A capability is defined as the ability of a company to combine resources to achieve a desired aim (Tarafdar & Gordon, 2007). The literature specifies capabilities by dividing them into the dimension's tangible and intangible assets. Tangible assets are fixed and/or physical within the company, e.g., technologies or raw materials (Tarafdar & Gordon, 2007; Wernerfelt, 1984). Intangible assets include among others values, brands, and knowledge (Grant, 1991; Tarafdar & Gordon, 2007). Intangible resources contain, e.g., knowledge assets, working culture, and relationships (Bharadwaj, 2000; Ross, Beath, & Goodhue, 1996; Tarafdar & Gordon, 2007).

The DevOps concept is an ongoing trend in IT, and an increasing number of firms implement DevOps. Therefore, capabilities of DevOps are needed for IT function that determine new business requirements as well as technological and organizational changes (Fitzgerald & Stol, 2017). This thesis contains several papers that present insights into capabilities (e.g., Papers 3 and 5) to foster an in-depth understanding of the DevOps concepts. Furthermore, Paper 2 presents a general introduction of the DevOps phenomenon, including key capabilities and principles.

In addition, the investigation of competencies for process innovations like DevOps is crucial for different reasons. First, the changes adopted by the business are dependent on IS and IT

(Bassellier, Benbasat, & Reich, 2003). Second, process innovations are IT-enabled to a certain degree, and that necessitates an understanding of IS competencies (Sambamurthy, Bharadwaj, & Grover, 2003).

Competency models are used to describe abilities, knowledge, attitudes, and traits that are necessary for the required performance of an organizational position or role (Mansfield, 1996; Matook & Maruping, 2014). Skills and knowledge underpin these competencies, which are not located only in the IT function (Peppard & Ward, 2004). Individuals have to adopt different roles during their working time. Research in human resource management describes a range of attributes that define a job role and differentiates employees from each other: skills, knowledge, behavior, and attitudes (Peppard & Ward, 2004; Staw, 1991). In general, competencies are ordered into two streams, performance-based and attributed-based (Klendauer, Berkovich, Gelvin, Leimeister, & Krcmar, 2012; Napier, Keil, & Tan, 2009). Performance-based perspective concentrates on a skillful practice to demonstrate the 'know-how' of an individual. Attribute-based perspective focuses on the extent of the individual's 'know-what', the required set of personal characteristics and knowledge (Napier et al., 2009).

The term competency is often used synonymously to skills, but needs to be differentiated. On one hand, skills highlight the ability to finish tasks and solve problems through the application of individuals knowledge. On the other hand, competencies represent the ability to use skills, knowledge, personal, and social capabilities to assess situations (Boehm, Stolze, & Thomas, 2013; Peppard, 2010). Skills outline the abilities and behavior needed to perform activities that are expected by the cross-functional DevOps team. Skill models help measure the performance of people to find out if they need personal development or to enhance effectiveness. Furthermore, they present a useful tool for recruitment and staff planning process (Lucia & Lepsinger, 1999; Matook & Maruping, 2014).

A broad skill set helps team members to develop new perspectives, ideas, and foster the information exchange outcome (Lee & Xia, 2010; Matook et al., 2016). The focus of this thesis is to present insights into the skills-as-attributes view, for obtaining the requirements of DevOps teams for successful practice. Paper 4 provides a better understanding of skillset for successful DevOps teams and considers skills as mandatory for high performance of DevOps teams. The skills-as-attribute method concentrates on specific and required skills, personal characteristics, and knowledge (Napier et al. 2009) of DevOps team members. This method helps to examine skills based on contextual aspects, measuring the relationship between the skill set and team outcome.

2.2.3 Control Theory

Control is defined as the actions used to align the behavior of an individual or group with its organizational goals (Ouchi, 1979; Wiener et al., 2016). Control is valuable for influencing stakeholder behavior and motivating individuals (Kirsch, 2004). In IS research, control is addressed from different perspectives. Considering different control perspectives supports examining control in DevOps settings. Therefore, explaining the different approaches of IS project control, which mainly focuses on software development, and providing overview of software operations control will help elucidate the control aspects in DevOps teams.

2.2.3.1 Software Development Control

This review of the IS project literature was guided by frameworks for IS processes (Cram, Brohman, & Gallupe, 2016a; Wiener et al., 2016).

This section reviews control literature (Kirsch, 1997; Ouchi, 1979; Wiener et al., 2016) in the area of software development projects to understand controlled-flexible development elements. The IS control literature concentrates primarily on managing IS projects in organizations (Kirsch, 1997; Mähring, 2002). Extant control literature differentiates between formal and informal IS projects modes (Jaworski, 1988; Kirsch, 1996). Formal control concentrates on performance evaluation regarding input, behavior, or outcome control. Informal control focuses on people and social aspects through clan, and self-control modes (Eisenhardt, 1985; Kirsch, 1996; Mähring, 2002). These forms of control are often combined in controller-controlee relationships (Ouchi, 1979) and used in control portfolios mechanisms for supporting management practices in different contexts (Kirsch 1997).

Table 2 summarizes the definition of formal and informal IS project control based on Wiener et al. (2016).

Control mode		Definition	Reference (examples)
Formal control	**Input control**	It is defined as the specification, monitoring, and handling of human, material, and financial project resources. The controller rewards or sanctions the controlee regarding the applied resources.	Jaworski (1988); Mähring (2002)
	Behavior control	It is defined as the influence on controlees to achieve a desired goal by stipulating concrete procedures and rules. The controller monitors tasks and the profitable compliance of members.	Heumann et al. (2015) Henderson and Lee (1992); Kirsch, Ko, and Haney (2010)
	Outcome control	It is defined as the interim and final output provision. The controller determines clear objectives while the controlee is rewarded or sanctioned for the output provided.	Harris, Collins, and Hevner (2009); Heumann et al. (2015); Maruping et al. (2009)
Informal control	**Clan control**	It is defined as the behavior of a group which is motivated by sharing norms, values, and common vision. Clan control is conducted through the controlee.	Chua et al. (2012); Kirsch, Sambamurthy, Ko, and Purvis (2002); Mähring (2002)
	Self-control	It is defined as the self-monitoring of intrinsic motivation, individual standards, and the goals of the controlee. The controlee defines his or her own objectives and the necessities to achieve it.	Kirsch (1996); Tiwana and Keil (2009)

Table 2. Formal and Informal IS Project Control (based on Wiener et al. (2016))

2.2.3.2 Software Operations Control

Control literature is not restricted to software development activities. Past examinations have predominately focused on IS project and software development control, but knowledge of control in software operations and maintenance is somewhat limited (Shaft & Vessey, 2006).

The focus of IT operations management and control regarding IT assets has changed a lot. In the late 1960s and 1970s the focus laid on mainframes for facilitating task automation and data processing. Specialized and high skilled employees were needed for operating this enterprise IT. Operational IT failures can appear in IT assets such as software, hardware, networks. Hence, there is a need for considering control over IT systems and architecture (Benaroch & Chernobai, 2017; Goldstein, Chernobai, & Benaroch, 2011). Digital transformation enables digital infrastructure (Tilson, Lyytinen, & Sørensen, 2010) and platform-based value creation (Tiwana & Konsynski, 2010; Wareham, Fox, & Cano Giner, 2014) that shapes operations control mechanisms.

The following section shows the findings of a comprehensive literature review to derive a logical configuration that describe control in software operations. The guidelines from Webster and Watson (2002) help to identifying suitable literature (additional information about the methodology are presented in section (3.2). The following keywords guided the research: *operations, architecture control, infrastructure control, systems control, maintenance, platform,* and *operations control.* For this study, the inclusion criteria consisted of (1) publications in peer-reviewed journals, such as the Senior Scholars' Basket of Journals and (2) studies that provide insights into the control of software operations, architecture and maintenance control. In addition to reviewing articles published in premier IS research journals, research presented at the last five years of key IS conferences, such as the International Conference on Information Systems were also reviewed, to include findings not yet published in journals. In all, 49 papers concentrating on software operations tasks and control were identified (see Appendix A). The following Table 3 depicts an excerpt of key papers that guided the research of this study.

Paper	Research method and aim	Main findings
Banker, Datar, and Kemerer (1991)	The paper presents a stochastic data envelopment analysis method. The research identifies factors of managerial control that have a significant impact on software productivity and provides a model for the estimation of software maintenance from an economic production perspective.	This research depicts a new model for software maintenance, including the multi-dimensionality of software lifecycle products. The model tests and enables the separability of software production functions and identifies individual software projects differences. The article presents the significance of software maintenance as an economically important managerial topic.
Nelson et al. (2000)	The authors use the qualitative methodology revealed causal mapping (RCM) to examine the software operations support expertise.	The paper identifies five constructs of software operations expertise: personal competencies, environment, motivation, IS group outcomes and IS policy. The article provides guidelines for recruiting and training software operations to support people. The authors argue that retaining experts inhouse is critical for

		software support and stress the importance of IS policies in the individual organizational context.
Shaft and Vessey (2006)	This article takes a quantitative research approach and explains considerable enhancements in comprehension that are associated with high-level performance on a modification task with cognitive fit theory.	The findings show that software comprehensions and modifications should be considered interrelated tasks with a complex interrelationship. High levels of comprehensions serve to enhance software-related tasks when software maintenance people's mental representation of the software and their mental representation of the modification task have the same kind of knowledge.
Bardhan, Demirkan, Kannan, Kauffman, and Sougstad (2010)	The paper shows a framework based on the findings of existing research that supports the evaluation of theory and methods for managing services-oriented systems.	The authors explore the issues and opportunities for new managerial knowledge for services-oriented systems through the design science, computer science, and IS and technology views. The robust framework gives insights into multiple roles of stakeholders and technological disruption.
Tilson et al. (2010)	The paper presents a research commentary on digital infrastructures to deliver an IS research agenda.	The authors highlighted that research in the area of digital infrastructure is rare and call for further investigations on platforms and heterogeneous environments. Control forms change from "command and control" toward "connect and coordinate," and control remains a focal research challenge.
Yoo, Henfridsson, and Lyytinen (2010)	The paper indicates a research commentary as an agenda for the new logic of organizing digital innovation and digital strategy and the creation of IT infrastructure.	The paper presents that digital product innovations are often set up with modular layered architecture which needs guidelines on how to strategically control the core components of digital platforms. The authors find that traditional control mechanisms are moving toward software-enabled control, which requires further investigation.
Edberg et al. (2012)	The paper depicts a grounded theory research to analyze how information technology experts define and use a methodology to maintain running software.	The paper models how software developers use a process for maintaining running software and proposes interrelationships between method and software maintenance. Hence, software maintenance tasks are defined by the individual experts, rather than a group of people or the organization.
Tiwana, Konsynski, and Venkatraman (2013)	The paper presents an introduction to a special issue and aims to show how organizations govern their IT tasks in emergent arrangements such as platform ecosystems.	The paper highlights an IT governance cube which presents further research possibilities along with IT governance. The cube addresses questions around what to govern? Who governs? And how is governed?

Trusson et al. (2014)	The paper combines critical discourse analysis, rhetorical analysis and quantitative dimensions to address the question how employees' experiences in sharing knowledge about IT service management are problematic and result in divergent perspectives.	The authors contribute to existing research by providing significant new evidence of IT service management processes and discourses. They identify the dysfunctional conflict between employees' practice and managers, and conclude that addressing this organizational issue lead to improvement of performance.
Benaroch and Chernobai (2017)	This paper presents an empirical quantitative study of changes when companies implement board-level IT governance when experiencing operational IT failures.	The authors contribute insights on operational IT failures and to IT governance literature. Through empirical tests, the paper shows the relationship among the stock market reaction to operational IT failures and changes in the level of board IT capabilities.

Table 3. Insights from Different Theoretical Dimensions of IT Operations (Own Depiction)

Investigations of operations control concentrate on infrastructure and systems control. Operations control refers to changing and managing day-to-day activities of the IT function. Looking at ITIL, it describes best practice for IT service management processes (Trusson et al., 2014). There is a need to integrate control mechanisms over IT systems (Benaroch & Chernobai, 2017) because evolving digital infrastructure (Tilson et al., 2010) and platform-based technologies (Ghazawneh & Henfridsson, 2013) require different kinds of control.

Extant studies investigate mechanisms for controlling software operations from different perspectives. Support control processes are identified as helpful for incident management (Nelson et al., 2000). Infrastructure control influences how users communicate with each other with the help of infrastructure components and assets (Yoo et al., 2010). System control refers to application-oriented control required to implement change and support software systems. Application-oriented changes are necessary for satisfying user demands during releases and after the deployment of new software code. Systems control includes monitoring software activity output (e.g., data flow, logging) for debugging purposes and for resolving emerging problems and incidents (Edberg et al., 2012).

To summarize the findings of existing investigations, an overview of the different IS software operations control modes are presented in Table 4.

Control modes		Definition	References (examples)
Systems control	**Release control**	It refers to the implementation of user demands that are necessary for software modifications installed with the help of a release.	Banker, Slaughter, Swatman, Wagenaar, and Wrigley (1994); Edberg et al. (2012)
	Change control	It refers to activities related to new software implementation when control of the	Baronas and Louis (1988); Duh, Chow, and Chen (2006); Lempinen and Rajala (2014)

		organizational and technological environment is needed.	
Support control	**Performance control**	It refers to monitoring the software activity output (e.g., data flow, and program slicing techniques) for debugging activities.	Banker, Davis, and Slaughter (1998) Shaft and Vessey (2006); Weiser (1982)
	Problem control	It refers to the resolution of emerging problems (e.g., incidents), where clear control guidelines are needed.	Nelson et al. (2000)
Infrastructure control	**Platform-based control**	It refers to a platform for measuring and monitoring data traffic. This information based on interfaces with other services, self-services, and communication exchange via technology.	Constantinides and Barrett (2014); Gregory et al. (2018); Koutsikouri, Lindgren, Henfridsson, and Rudmark (2018); Yoo et al. (2012); Yoo et al. (2010)
	Software-enabled control	It refers to the provision of digital innovations enabled through modern modular infrastructure. Knowledge exchange happens with humans and other technologies.	Hanseth and Monteiro (1997); Yoo et al. (2010)

Table 4. Overview of IS Software Operations Control (Own Depiction)

This section presents an overview of software development and operations control builds upon existing literature in the area of control. The dissertation aims to examine the control elements of cross-functional teams used to align developers and operations people and accomplish product orientation Thus, this dissertation theoretically develops the idea that control perspectives from software project as well as software operations are complementary to each other. Furthermore, this dissertation investigates organizations during their DevOps transformation.

2.2.4 Business-IT Alignment

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). IS research has focused on alignment, which has been a top concern among IS executives for decades (Gerow, Grover, Thatcher, & Roth, 2014a; Henderson & Venkatraman, 1993; Reynolds & Yetton, 2015).

The Strategic Alignment Model (SAM) indicates how firms align elementary dimensions that support realizing the potential of IT (Henderson & Venkatraman, 1993). The four dimensions are business strategy, IT strategy, organizational infrastructure and processes, and IT infrastructure and processes. Existing research depicts that business-IT alignment positively influences the potential of IT (Gerow et al., 2014b) as well as firms performance (Chen, Mocker, Preston, & Teubner, 2010). The SAM depicts that IT can be more effectively managed if choices made across the four dimensions. The SAM implies that IT and business components are categorized by three levels: intellectual alignment, operational alignment, and cross-domain alignment (Gerow et al., 2014a; Henderson & Venkatraman, 1993). Figure 2 demonstrates the SAM.

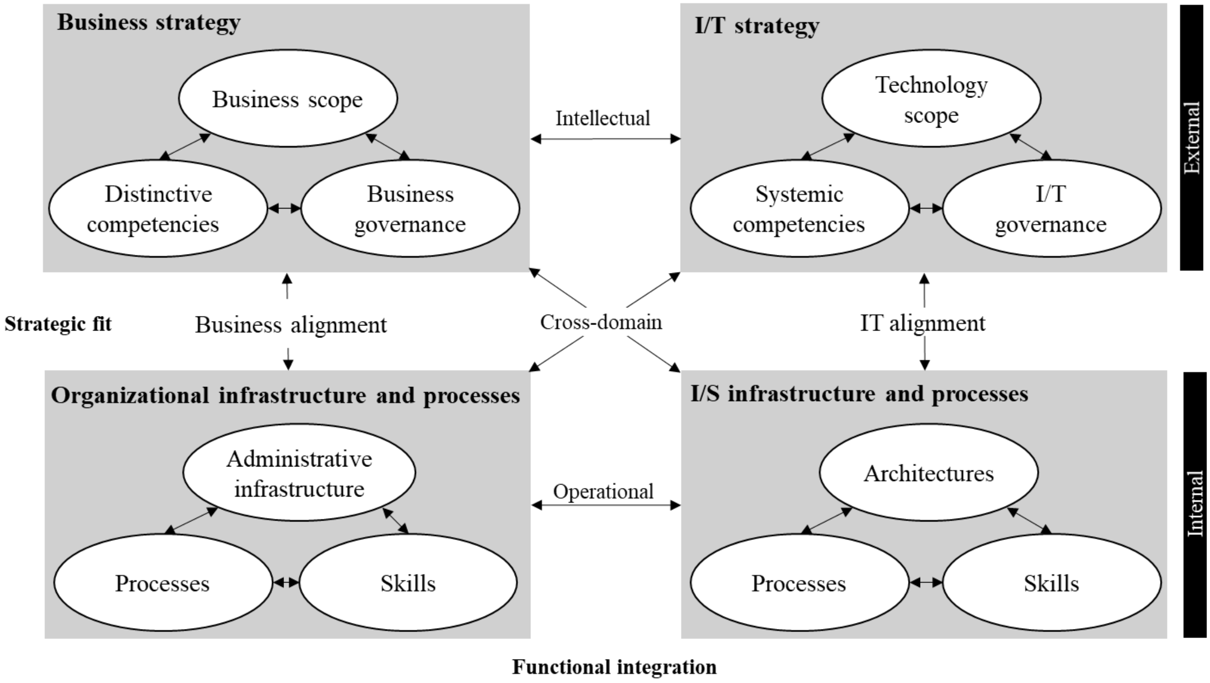


Figure 2. The Strategic Alignment Model (based on Henderson & Venkatraman, 1993)

Intellectual alignment takes place at the executive and management levels and includes governance, business, and technology scope as well as competencies (Henderson & Venkatraman, 1993). Operational alignment concentrates on infrastructure, business, and IT processes focusing on cooperation within the same internal level of business and IT (Gerow et al., 2014a), with the goal of aligning architecture, processes, and skills. Cross-domain alignment refers to the degree of fit and integration between IT and business strategy as well as IT and business infrastructure and processes (Chan & Reich, 2007). The SAM is a well-known model and prior research builds up. For example, Reynolds and Yetton (2015) provide a new model of business-IT alignment. This alignment model presents explanations of IT value creation over the investment lifecycle between and within corporate level and strategic business units.

2.2.4.1 Operational Alignment and Intra-IT Alignment

DevOps is a phenomenon that appears mainly within the IT function. Extant research depicts that operational alignment literature is suitable for explaining alignment within the IT function (Onita & Dhaliwal, 2011). Henderson and Venkatraman (1993) describe that operational alignment influences organizational infrastructure and processes including topics like organizational structure, work- and information flows, and operational capabilities required for implementing strategic aims. IT infrastructure and processes involve definitions, hard- and software technologies, data flows, and work processes that are necessary for the smooth operation of IT infrastructure and workflows.

Intra-IT alignment like aligning operational activities within the IT functions remains challenging. Developing capabilities and collaborations within internal IT units is increasingly necessary to cope with contemporary challenges and to achieve high IT performance (Dhaliwal et al., 2011). The main goal of intra-IT alignment is to achieve a cooperative and integrative form to operate congruently within the IT function (Dhaliwal et al., 2011; Onita & Dhaliwal, 2011). One type of operational alignment is intra-IT alignment, which needs scholarly attention (Dhaliwal et al., 2011). Two articles describe alignment within IT functions as depicted in Table 5.

Paper	Description
Onita and Dhaliwal (2011)	This article examines a development-testing alignment model. The model suggests 20 topics, such as changes in governance, resources, and processes that should be considered to achieve better alignment. The authors find that changes in specificity of scope, processes, resources, governance, availability, and competencies foster alignment between IT subunits.
Dhaliwal et al. (2011)	The paper investigates the role of the relationship between software developers and software testers in achieving alignment within the IT function. They suggest mechanisms that enhance alignment between the two IT subunits to prevent misalignment. Their findings present insights that relational but not structural dimensions influence intra-IT alignment.

Table 5. Examinations in Intra-IT Alignment (Own Depiction)

2.2.4.2 Misalignment within IT

Misaligned IT subunits can adversely affect general cohesion of the complete IT function, as it seeks to meet the aims of both business and IT (Dhaliwal et al., 2011). The two IT subunits, development and operations, pursue fundamentally different goals. While development seeks novelty and innovation, operations is concerned with stability (Markus & Keil, 1994). Differences include the planning and types of activities as well as the range of knowledge that is necessary to deal with these tasks (Edberg et al., 2012; Swanson & Dans, 2000). These gaps between the subunits can lead to misalignment (Dhaliwal et al., 2011; Onita & Dhaliwal, 2011). The components of misalignment between software development and operations are briefly introduced.

Achieving intra-IT alignment between development and operations is a fundamental challenge in IT functions (Laney & Jain, 2016; Onita & Dhaliwal, 2011). Software developers are typically not involved in processes are used after the software component is deployed. Nevertheless, there are dependencies between development and operations tasks after deployment, which is needed to be considered by both of them (Edberg et al., 2012; Kemerer & Slaughter, 1999).

Misalignment between development and operations can appear in different forms (Wei, Wang, & Ju, 2005). For example, one aim of development is to quickly provide new software features, while operations seek stable running systems with few changes (Edberg et al., 2012). Achieving alignment between these two opposing goals requires assimilation among the processes, skills, and architecture of an organization (Henderson & Venkatraman, 1993). Developers act proactively, primarily in project settings, to provide new functionalities to the customer. In contrast, operations are usually reactive and typically begins when a problem appears in the running software system (April, Huffman Hayes, Abran, & Dumke, 2005).

The perception of development and operations activities differs at the executive level. Managers are interested in new and innovative projects that typically require new developments. Operations work is almost focused on significant interests in a smooth-running system. If problems appear, they should be solved and fixed quickly (Edberg et al., 2012). Achieving alignment between the different perception leads to one common strategic orientation.

As introduced above, there are several possibilities for misalignment when DevOps is integrated into existing IT functions. In DevOps teams both, software development and operations are interrelated. Hence, there is a gap in research that concerns how these misalignments are solved or avoided.

3. Research Methodology

3.1 Philosophical Perspective

This thesis presents investigations with several theoretical perspectives. Chapter 2 provides an overview of the main theories used in this dissertation. The application of different research methods helped to deliver a big picture of the DevOps phenomenon and improve the theoretical understanding of antecedents, effects, and impact of DevOps integration.

Every research has a philosophical perspective (Urquhart, 2012). The philosophical perspective presents underlying assumptions guiding the research conducted. Such philosophical perspectives are related to epistemology, which refers to the assumptions of how knowledge is acquired (Myers, 1997). Orlikowski and Baroudi (1991) identified three basic underlying epistemology categories: positivism, interpretivism, and critical as depicted in Table 6.

Research philosophy	Epistemology
Positivism	Positivist research is deductive and evaluates unilateral, causal relationships. The core of this philosophy is the deriving of hypothesis from theory and the testing for verification and falsification. Positivism depicts the necessity of facts and distinct values of scientific knowledge (Orlikowski & Daniel, 1991; Urquhart, 2012; Walsham, 1995).
Interpretivism	Interpretive research examines the phenomenon in social settings. An in-depth analysis of the field is necessary to derive constructs. Investigators follow the aim to construct interpretations of their investigations and meanings (Walsham, 1995).
Critical	Critical research beliefs that knowledge is grounded in historical and social practices. Critical research is often combined with longitudinal studies. Typically, there is no theory independent collection of evidence to prove or disprove an existing theory (Orlikowski & Baroudi, 1991; Urquhart, 2012).

Table 6. Research Philosophies (Own Depiction)

The philosophical position of this dissertation has an underlying interpretive epistemology because it interprets social practices (Walsham, 1995).

Eisenhardt (1989) investigates the issue of epistemology in the context of organizational research and presents three possibilities for using theory:

- As a preliminary guide to research design and data collection
- As a part of an iterative procedure of data collection and data analysis
- As a final outcome of the research

Most of the papers in this dissertation present case study research and use one of the three possibilities. Interpretive research studies preserve considerable openness to the field data and initial assumptions and theories are often modified. Data collection and analysis are iterative processes to broaden, rework, or cancel initial ideas (Walsham, 1995).

Prior research provides insights into interpretive studies and the Grounded Theory approach (Glaser & Strauss, 1967). These investigations recommend exploring theory directly from the data and constructing new theories in comparison with existing theory. Orlikowski (1993) shows a Grounded Theory research as the basis of an interpretive case study to demonstrate the use and adoption of computer-aided software engineering (CASE).

Interpretive case studies involve three issues, the role of researcher, evidence from interviews, and reporting methods:

The role of the researcher describes the view and complex processes of a human in the role of a researcher. During in-depth case studies, researchers become a temporary member of the research field and their observations are in part subjective. The researcher is positioned along the spectrum between 'involved researcher' and 'outside observer' (Walsham, 1995). In this thesis, the researcher adopted a largely outside observer role as researchers. Although the researcher was personally involved in every interview and study participants were often visited onsite, sufficient distance was maintained to ensure that case study participants viewed the researcher as an outsider (Walsham, 1995).

Evidence from interviews describe the different forms of data collection. For example, case studies can use different sources: documents, interviews, archival notes, observations (direct or participant), and physical artifacts. Expert interviews are a popular data sources in case study research (Walsham, 1995; Yin, 2018). As an outside observer, the researcher can have access to actions and events for a temporary range and collect interpretations from the participants. To conduct interviews effectively, the researcher must have good social skills and personal sensitivity (Walsham, 1995).

Reporting methods describe facts and interpretation methods to explain how the researcher derives the findings. In an interpretive case study, the collection of data includes details of the research site, reasons for conducting the research, participants' characteristics, secondary data, and period of data collection. Explaining data analysis requires information on data recording and transcriptions as well as details about the data analysis method (Walsham, 1995).

3.2 Literature Review

The guidelines suggested by Webster and Watson (2002) and Wolfswinkel, Furtmueller, and Wilderom (2013) offer methodologies for conducting literature reviews. Literature reviews serve as summaries on current states of research and synthesize existing knowledge. Furthermore, they identify research gaps and opportunities for unexplored research questions. Reviewing literature is a necessary process in scholarship that ensures connecting new research to the existing body of knowledge. Systematic literature reviews aim to synthesize existing research and identify possibilities for further research. Following the advices from Webster and Watson (2002), the definition of inclusion and exclusion criteria or prior research is one of the first steps starting a literature review.

Wolfswinkel et al. (2013) indicate a Grounded Theory approach for conducting literature reviews. This approach builds upon Webster and Watson (2002) for systematically reviewing existing literature. The work of Wolfswinkel et al. (2013) aims to facilitate the theoretical progress of reviewing literature by offering a more systematic and rigorous approach rooted in Grounded Theory. By means of repeating processes of data collection and analysis, this approach aims to build and correlate concepts to develop new or extend prior theory (Strauss & Corbin, 1990). Five stages facilitate the iteratively evolvement of *"a theory-based or concept-centric yet accurate review"* (Wolfswinkel et al., 2013, p. 47) and are presented in detail below:

Define: The define stage includes the definition of inclusion and/or exclusion criteria, the identification of the field of research, the determination of appropriate sources, and the decision of specific search terms.

Search: The search stage displays the identified sources for searching through existing sources (e.g., data bases and journals) and documenting findings and insights.

Select: The select stage concentrates on identifying doubles and papers that do not fit the research criteria. Typically, the titles, abstracts, and keywords are read to select appropriate papers. Forward and backward searching supports the selection process.

Analyze: The analyze stage includes open, axial, and selective coding steps. These coding steps are the key principles of Grounded Theory provided by Strauss and Corbin (1990). Researchers start with random papers and read as well as highlight relevant text for their search. Afterward, the researchers reread the excerpts. Through this iterative procedure several concepts and meta-insights start to appear. After the open coding process, a higher-order conceptualization into categories is necessary. These categories can have properties that demonstrate the differences between categories and concepts. Hence, through axial coding interrelations between categories and sub-categories are identified. Eventually, selective coding integrates and polishes the identified categories. This coding form supports the building of relationships between the key categories.

Present: The present stage represents and structures the content of the articles. This stage supports the development of a logical flow for presenting the findings from the literature review, commonly using tables or visual means to help the reader process the findings.

For this doctoral thesis, several guidelines were triangulated to achieve a foundation for further investigations (Webster & Watson, 2002; Wolfswinkel et al., 2013). The literature review based on Wolfswinkel et al, (2013) as presented in Paper 1 aims to clarify the organizational learning set-up in cross-functional agile IT teams. The examination builds upon the existing concept of organizational learning (human, social, and organizational) and add a technology dimension. These dimensions provide the groundwork for further investigations depicted in this thesis.

3.3 Qualitative Research

Qualitative research supports investigators for studying cultural and social phenomena. This research methodology includes case studies, action research, and ethnography. Qualitative research studies enhance the development of mechanisms that lead to specific events and outcomes (Maxwell, 2012). Qualitative studies are a suitable methodology when the subject cannot be easily measured and support the determination of measurement instruments for quantitative studies (Kaplan & Maxwell, 2005). Qualitative research aims to uncover theoretical boundaries, drive theoretical insights and provide an all-encompassing story because with *"qualitative research, we are trying to see the forest through the trees"* (Bansal & Corley, 2012, p. 511).

Qualitative research investigates significant gaps in current theoretical conversations. The major aims of qualitative research are theory development or extension. A unique feature of qualitative research is the credible grounding of a phenomenon through uncovering it in a nonlinear and unique way. It is challenging for qualitative researchers to illustrate their data in a synthesized way. Nevertheless, data visualization is a necessary condition to achieving a high level of traceability and to provide the context for the reader (Bansal & Corley, 2012).

Qualitative research is widely used for exploratory research to derive new findings and theory from inductive approaches and to provide a better understanding of an under-investigated phenomenon (Venkatesh, Brown, & Bala, 2013).

The findings of Papers 3, and 5-9 base on multiple case study research. This research design allows analyzing the real-life phenomenon of DevOps teams within multiple organizations. Generally, when research questions ask "how" and "why" about a contemporary phenomenon or set of events underscore the appropriateness of case study research (Yin, 2018).

Paper 2 presents a general overview of the DevOps phenomenon based primarily on case study research and literature reviews published in the past years. Paper 4 presents the results of qualitative research conducted following workshop methodology involving DevOps experts and single expert interviews. The following section 3.3.1 describes the workshop methodology.

3.3.1 Workshops

Conducting workshops enables the elaboration of a 360-degree view on the topic. A workshop is a suitable method for examining a nascent phenomenon. This method allows group discussions and an iterative way of verifying results (Boehm et al., 2013). The goal of workshops is to discover issues and ideas through a collaboration between research and practice. Workshops enable the management of complex situations. Organizations can be observed and fruitful discussions and visualization of findings lead to the required results (Howard, Vidgen, & Powell, 2004).

Using workshops as a research method needs an investigator as a process supervisor. On the one hand, the investigator facilitates the discussions and explains the research aim. On the other hand, biases need to be avoided. The workshop should begin with a short introduction about the topic and the participant must agree on the subject area. The opinion of the participants is collected, and researchers are in the background and only take part in the discussion when necessary, e.g., when discussion strays from the subject area (Howard et al., 2004).

Whiteboards and pinboards visualize the result of the workshops. The participants write their ideas on memos. These are pinned on the boards or fixed at the whiteboard and preserved digitally. Furthermore, at the end of the workshop, a discussion of the findings supports a common understanding. After the workshop, the findings are distributed to the participants and individual feedback is collected from the participants (Howard et al., 2004).

Paper 4 presents an ideal skill set for DevOps teams as collected in a workshop with IT experts from a consultancy company. The findings of this workshop present a comprehensive view of key skill categories with detailed descriptions and skills that exemplify the skill categories.

3.3.2 Case Study Research

Case study research is traditionally associated with qualitative methods. Case study approaches are defined as *"an empirical inquiry that investigates a contemporary phenomenon in depth and within its real-life context"* (Yin, 2009, p. 18).

A case study is defined as an empirical research method with the following scope:

- the examination of a contemporary phenomenon in depth and within its real-life context
- the boundaries between context and case are not obviously clear

The methodology characteristics are defined as the features of a case study when:

- it deals with technically distinctive states in which there may be many more points of interest than data points
- it profits form extant theoretical development of initial propositions to guide the research design, data collection, and data analysis
- it leaves on multiple sources of evidence that need a comprehensive set of data to cover in triangular fashion (Kaplan & Maxwell, 2005; Yin, 2018).

Case study research is traditionally associated with qualitative methods. This dissertation uses case studies and takes an inductive research approach that are applicable for analyzing new topics (Eisenhardt, 1989). The overall aim is to develop theories from qualitative methods (Miles & Huberman, 1994), using the case study design (Yin, 2018), and Grounded Theory (Glaser & Strauss, 1967).

The case study strategy aims to provide a deeper understanding of dynamics in single settings. There are two different forms of case studies, single- and multiple-case studies that have either a holistic (single unit of examination) or embedded (multiple units of examination) approach. These forms serve to achieving different research aims (Yin, 2018). For example, Gregory, Beck, and Keil (2013) carried out a longitudinal single case study within a financial service firms to examine client-vendor relationships in an information systems development offshoring project. Slaughter et al. (2006) conducted a multiple-case study with nine different firms to find out how firms align components of software products, processes, and strategy in internet software application development between business and IT.

The inquiries of this thesis consist of holistic multiple case studies. That means several companies participate in the case study. In every company, one DevOps team was in focus of the investigation. Multiple-case studies are more robust because the evidences are more compelling than in single-case study designs. Multiple case studies are used to replicate a result or identify contrasting conditions (Yin, 2018).

3.3.2.1 Research Site

The multiple-case study presented in this thesis includes incumbent firms in Canada, Germany, the USA, Africa, Asia, and Australia. The case study participants range in size from 50-100 to >100,000 employees and are from several industries (see Table 7.). All examined firms have integrated the DevOps concepts to a significant degree and have internal DevOps teams responsible for at least one digital product. Several cases were identified at practitioner conferences at which these companies held presentations and were positioned as outstanding examples for DevOps and identified by others as role models. A key criterion for participating in the study was that the organizations started to integrate DevOps principles (e.g., collocating people in teams) at least six months ago. This research design includes one cross-functional team per interview and several interviews per firm. While all firms have integrated DevOps-oriented teams, their industries, products, and customers are different.

The international context influences DevOps through differences among time zones, local differences, and culture (Persson, Mathiassen, & Aaen, 2012). A transnational focus for this study has been chosen, because the movement from traditional IT functions toward DevOps is global and because international case studies serve as role models for other firms. In addition, this dissertation focuses on cultural differences in skills, control, and alignment. These differences help to identify a broad set of solutions based on new and comprehensive theoretical models. To this end, it was valuable to draw on outstanding and extreme international examples of integrated or partly integrated DevOps teams. Table 7 presents an overview of the research sites.

	Cases and context	IT function	Interviewees	DevOps team description	Year *
			Germany		
1	**Media** A subsidiary of a leading television station with more than 50 employees. The company provides content for websites and videos of other firms.	The complete IT function is managed with the help of DevOps.	2 interviews (2 executives)	12 people with mainly development backgrounds (full stack) are responsible for providing data to other services. The team uses Scrum principles.	2016
2	**Pet** A pet supplier business with more than 500 employees. The company sells their products in an online shop.	The complete IT function is reorganized with the help of DevOps.	2 interviews (team lead, senior engineer)	6 people with mainly development backgrounds are responsible for an online service of the website (customer feedback service). The team uses Scrum principles.	2016
3	**Furniture** A leading online furniture store with more than 1,000 employees. The company sells their products in an online shop and in showrooms.	The complete IT function is managed with the help of DevOps.	2 interviews (CTO, senior engineer)	5 people with mainly development backgrounds are responsible for an online shop service. The team uses Kanban principles.	2016
4	**Health Care** A leading health care technology provider with more than 50,000 employees. The company operates worldwide in different sectors and industries.	The IT function is mainly traditional oriented with some DevOps teams in IT function.	2 interviews (2 executives)	10 people from Germany and India with development and operations backgrounds are responsible for a medical device in hospitals. The team uses Scrum principles.	2016
5	**IT Provider** A global IT provider and consultancy firm with more than 100,000 employees. The company consults different industries.	The IT function is mainly traditional oriented with some DevOps teams in IT function.	2 interviews (team lead, senior consultant)	3 people with mainly development backgrounds are responsible for a payment massage system. The team uses different agile elements.	2016
6	**Bank** An international Bank with more than 2,500 employees in Germany. The company is organized by stores, online banking and apps.	The IT function is mainly traditional oriented with some DevOps teams.	2 interviews (team lead, senior engineer)	23 people with different backgrounds are responsible for a big data service. The team uses Scrum principles.	2016

7	**Holiday** A consumer portal for travel and leisure activity bookings. The company has more than 250 employees and provides their services via a web site.	Their IT function has a DevOps team for their platform and some DevOps-oriented teams.	2 interviews (team lead, DevOps engineer)	5 people are responsible for an online platform service. The team consists of operations people and one DevOps engineer and uses Scrum principles.	2017
8	**Telecommunications** A big telecommunications supplier for a city in Germany. The company has more than 500 employees and various sales channels, including shops and websites.	The company started transforming some projects to DevOps.	2 interviews (2 executives)	8 people are responsible for a customer online portal. The team consists of mainly software developers and uses Scrum principles.	2016/2017
9	**Internet** An online company that provides software for digital services and digital media via apps and online shops. The company has between 10 and 50 employees.	The complete IT function is managed with the help of DevOps.	2 interviews (CTO, senior software engineer)	7 people with different backgrounds are responsible for an online platform-service. The team uses Kanban principles.	2017
10	**Consultancy** A consultancy firm with focus on agile working methods and individual software solutions. The company has more than 50 people.	The company started transforming some projects to DevOps.	2 interviews (CTO, senior software engineer)	4 software developers are responsible for an industry 4.0 prototype services. The team uses Scrum principles.	2017
11	**Energy** A company that provides innovation and software solution mainly for water and energy industry. The company has more than 5,000 employees.	The company started transforming some projects to DevOps.	2 interviews (team lead, IT specialist)	20 software developers are responsible for a customer self-service website in the energy sector. The team uses Scrum principles.	2017
12	**Food** A leading food and convenience retail company with more than 100,000 employees. The company is organized by stores and online shopping sales channels.	The IT function is transforming from traditional set-up to DevOps orientation.	6 interviews (4 team members, product owner, agile coach)	7 software developers are responsible for an app for managing a delivery service. The team uses mainly Scrum principles.	2018

13	**Finance** A leading financial banking institution with more than 100,000 employees. The institution offers various types of banking products and online banking.	The IT function is mainly traditional oriented with some DevOps teams in IT function.	4 interviews (former group manager, group manager, 2 team members)	15 people with mainly operations backgrounds are responsible for a securities management system. The team uses an agile-traditional hybrid approach.	2018
14	**Insurance** A leading insurance company with more than 10,000 employees. The company offers various types of insurance through different sales channels.	Their IT function has a DevOps team for their platform and other services are traditional oriented.	2 interviews** (executive, manager, team lead)	10 people with mainly development backgrounds are responsible for an internal delivery platform. The team uses Scrum principles.	2018
15	**Retail** A leading retail company with more than 50,000 employees and different sales channels (e.g., shops and online).	The complete IT function is managed with the help of DevOps.	2 interviews (team lead, team member)	9 people with mainly development backgrounds are responsible for an online shop. The team uses Scrum principles.	2018
16	**Warehouse** A warehousing business with more than 20,000 employees and stores and online shops as sales channels.	Their IT function has a DevOps team for their platform.	3 interviews (team lead, 2 team members)	6 people with mainly operations backgrounds are responsible for an online shop and manages the basis platform. The team uses Kanban principles.	2018
17	**Grocery** A leading foods and retail chain company with more than 100,000 employees mainly in Europe. Its sales channels are stores and an online shop.	The company started transforming some projects to DevOps.	3 interviews** (division manager, 2 managers, team member)	6 people are responsible for configuration management tool and supporting other teams. The team uses Scrum principles.	2018
18	**Service** An Internet company with more than 1,000 employees. Helps end customers identify, compare, and buy products.	The complete IT function is managed with the help of DevOps.	5 interviews (team lead, 2 team members, 2 site managers)	4 people are responsible for product providing web services to end-user. The team uses Kanban principles.	2016/2018
19	**Travel** A well-known online travel agency with more than 1,000 employees in Germany. Serves customers mainly via an online shop.	Their IT function has a DevOps team for their platform.	4 interviews (team lead, 3 team members)	6 people are responsible for an online shop platform. The team uses Scrum principles.	2018

USA					
20	**Conglomerate** A diversified industrial corporation with more than 100,000 employees operating worldwide in different sectors and industries.	The IT function is mainly traditionally oriented with some DevOps teams in IT function.	4 interviews (manager, senior IT director, 2 DevOps engineers)	7 people are responsible for frontend service of a portal for customer request for new technology. The team uses Kanban principles.	2018/2019
21	**Telecommunications** A worldwide operating telecommunication provider with 1,500-5,000 employees which provides services and systems to their customers.	The IT function is transforming from a traditional set-up to DevOps orientation.	4 interviews (Senior Vice President, 2 executives, principal architect)	4 people are responsible for cloud-based platform supporting other teams. The team uses Kanban principles.	2019
Canada					
22	**Insurance** One of the leading insurances that offers insurance services in many different areas such as car, home, life and more provided by 50,000-100,000 employees.	The IT function is highly traditionally oriented and is starting to integrate DevOps teams.	3 interviews (Assistant Vice President, senior manager, DevOps Manager)	12 people are responsible for integrating automation processes within the company and other teams. The team uses Scrum principles.	2018/2019
23	**Technology Provider** A technology firm with 100-250 employees which provides tools for software development and delivery for automation and integration of these processes.	The IT function has some internal DevOps teams and some traditional services.	4 interviews (team lead, 3 team members)	6 people are responsible for developing and managing integration hub for external customers. The team uses Kanban principles.	2018/2019
Australia					
24	**Research** A research and consulting institution with more than 250 employees. The company conducts research in different areas such as infrastructure and health.	The IT function has innovation teams and some DevOps teams.	2 interviews (team lead, DevOps engineers)	9 people are responsible for a platform and managing network data structures. The team uses Scrum principles.	2019

		Africa			
25	**Finance** A leading bank in Africa with more than 50,000 employees started agile transformation and is now turning toward DevOps. The institution offers various types of banking products, apps and online services.	The IT function started agile and is now turning toward DevOps with the help of Scaled agile framework (SAFE).	6 interviews (coach, 2 solution engineers, 2 enterprise architects, team lead)	11 people are responsible for digital credit card services. The team uses Scrum principles.	2019
		Asia			
26	**IT consultancy** A leading IT consultancy firms with more than 100,000 employees. The institution is working worldwide in different projects and industries.	Work in projects to integrate DevOps principles to their customers.	2 interviews (project managers, DevOps specialist)	54 consultants and project members from the customer are working on the transformation from agile development toward DevOps teams.	2019
Additional seven interviews with thought leaders and industry experts					
5 prominent DevOps proponents who have significantly impacted the practice community through international lectures and publications (all from the USA). 1 system administrator with over 35 years of experience in a global technology company in the USA. 1 manager with extensive DevOps research and practical expertise in Australia.					

*of interview
** some interviews were held with two people

Table 7. Global Multiple Case Study Participants and Context (Own Depiction)

3.3.2.2 Data Collection

The described global multiple case study includes data from 26 cases and 75 single interviews were collected (see Table 7). In addition, several expert interviews with thought leaders from the US and one industry expert from Australia were conducted. The data from these global distributed cases as well as additional expert and thought leader interviews are fundamental for Papers 3, 5-9 of Part B1 and B2 of this thesis.

Consistent with the predominantly described inductive, interpretative research approach, this thesis was guided by several analysis methods, but mainly applied Grounded Theory Methodology (GTM). Constant comparative analysis was used to identify concepts and categories (Glaser & Strauss, 1967; Sarker & Sarker, 2009; Walsham, 1995). Table 8 presents a summary of the key guidelines and methodologies applied in this dissertation by following the steps presented by e.g., Sarker and Sarker (2009) and Eisenhardt (1989).

Method	Description	Example
Entering the research field and choice of organization		
Selecting appropriate case study partners	The aim was to identify and examine representative case study partners with varying degrees of DevOps integration.	A leading food and convenience retail company with more than 100,000 employees was mentioned in several publications as an outstanding example for DevOps integration.
Entering the research field	Firms were contacted via e-mail, telephone, and personally to achieve a high level of credibility. Then the research ideas were presented and anonymity guaranteed to build a trustworthy relationship (Myers & Newman, 2007).	An e-mail introduction of the research was sent to the executive by e-mail for further distribution in the company and initial meetings and calls were set up. Well-prepared interviews and guaranteeing anonymity and verbal confidentiality helped to foster trust (Myers & Newman, 2007).
Data collection		
Identifying interviewees	Pre-discussions or recommendations of identifying other interview partners following the ‘snowballing’ principle (Sarker & Sarker, 2009).	Different roles, job levels (DevOps managers and team members), software products, and countries helped to achieve depth and breadth in the studies.
Conducting interviews	For this thesis the focus laid on semi-structured interviews with open-ended questions lasting approximately one hour of each.	All interviews were held by the author of this dissertation either personally, via telephone, or via video conference.
Researcher’s involvement	The researcher had persistent interactions with the subjects and prolonged engagement with the participants.	The data collection was undertaken from the end of 2016 until mid-2019 and includes 75 interviews in 26 organizations and 15 on-site visits. Furthermore, seven DevOps experts and thought leaders, including keynote speakers at practice conferences were interviewed.

Neutral attitude	Researcher aims to maintain distance with a non-judgmental attitude while the interviews addressed critical topics (Patton, 1990; Walsham, 1995).	Sometimes critique or displeasure was addressed by an interviewee on a topic, was managed by sympatric, trustworthy and patient listening.

Table 8. Key Methodologies and Guidelines (Own Depiction)

3.3.2.3 Data Analysis

GTM gives insights into theoretical sampling, rigorous coding, memo writing, and constant comparison when analyzing data (Glaser, 1978). In line with GTM, the concepts related to the research phenomenon emerged over time (Urquhart, 2012). Iteratively refining concepts in a cross-validation process during the research process ensures reliability (Yin, 2018).

After each interview, the writing of memos helps to synthesize new findings and identify issues. These memos iteratively influence the refinement of interview questions and approaches (Urquhart, 2012). Strengthening the GTM is possible by triangulating the data with secondary data. Using multiple data sources supports case study validity and helps to mitigate potential bias (Yin, 2018). Focusing on publicly available data, such as company websites, blogs, and annual reports, as well as insights collected at conferences, supported the research process. Finally, integrating theoretical findings with the theory and domain literature fosters the development of a model for aligning development and operations (Wiesche, Jurisch, Yetton, & Krcmar, 2017).

3.3.2.3.1 Grounded Theory Method

For this thesis, data was collected in in-depth field investigation (Urquhart, 2012), and a classic conceptualist-grounded theorizing technique was adopted. The generalization of the theoretical concept is extended through the inductive concepts generated by extant theory as recommended by Glaser and Strauss (1967). Summary of key guidelines from Grounded Theory is described in Table 9.

Method	Description
Constant comparison	Is the process of continuously comparing the instances of data to identify categories. Codes, data, instances, and categories are constantly compared to achieve rigorous scrutiny and contributes to theory development. In vivo coding and theoretical concepts refine the emerging ideas constantly through open and selective coding (Charmaz, 2000; Urquhart, 2012).
Iterative conceptualization	The level of abstraction increases through iterative conceptualization for relating the categories with each other. Theoretical coding supports the building of relationships among the concepts. The process of theory building includes the understanding of links and patterns (Miles & Huberman, 1994; Urquhart, 2012).

Theoretical sampling	Theoretical sampling enables theory development and guarantees that the theory is grounded in data. The research question is polished through successive theoretical sampling corresponding to the emergent theory and the research problem gets clearer through the process of analysis (Glaser & Strauss, 1967). The theory is extended to a higher degree of generalizability and scope because emerging concepts from theory support the data sampling process to build and extend the theory (Urquhart, 2012).
Scaling up	The newly developed theory needs to be scaled up to an adequate level of abstraction. The aim is to relate the new theory to other theories in the research field. It supports the generalization of theory by broadening low-level theories to higher-level approaches. A recommendation is to have one or two core categories with a reasonable degree of abstraction (Glaser, 1978; Urquhart, 2012).
Theoretical integration	Provides guidelines for relating emerging theories to existing theories that are similar in the research areas. Formal process models, analysis and structures are supporting the process of theory integration. The theory is grounded by systematically integrating emergent theories into existing literature (Glaser, 1978; Urquhart, 2012; Urquhart, Lehmann, & Myers, 2010).

Table 9. Summary of Key Guidelines for Using Grounded Theory (based on Urquhart et al. (2010))

3.4 Summary

Table 10 presents an overview of the used research methodologies in this doctoral thesis and the corresponding papers. Furthermore, the table depicts which papers applied the research method described below.

Scientific insights
Global multiple case study research
The main findings based on the global multiple case study as primary data and secondary data from company presentations and Internet publications such as blog articles. Suitable case study participants for a theoretical sample were selected (Eisenhardt, 1989). Table 7 presents a comprehensive depiction of the global case study. The case study results are presented in the Papers 3, 5-9 of Part B1 and B2.

Description	Results
Literature reviews	
Systematic literature review	
Gathering insights about the key capabilities of cross-functional teams and core elements of software development, project management, and operational control mentioned in IS research (e.g., Senior Scholars' Basket of Journals) with the help of guidelines presented by Webster and Watson (2002).	• Current state of knowledge about control in software development and software operations. • Identification of organizational learning capabilities in cross-functional teams (Paper 1) and core elements and descriptions of operations control (part of Papers 7 and 9).
Multivocal literature review	
This form of review provides guidelines to combine academic and practice literature (Garousi, Felderer, & Mäntylä, 2016). This approach is appropriate for DevOps because there is limited academic research into it, yet it is frequently discussed in practice-oriented literature.	• Closing DevOps knowledge gap through examination of both practice-oriented and academic literature. • Composing explanations about DevOps and identifying key characteristics. • Identification of challenges and consequences of applying the DevOps concept (Papers 2-9)
Academic conferences and presentations	
Participation and presentations at several academic conferences in North America, Asia, and Europe (e.g., AIS conferences) and research presentations at universities in Germany, the USA, and Australia followed by research discussions.	• Exchange of experiences and knowledge with academic colleagues • Participations at doctoral consortiums • Collecting insights for entering the field • Discussion of ideas for further research opportunities (Papers 1-9).
Practice insights	
Workshop with IT consultants	
Organization of a workshop with an IT consulting firm with DevOps domain experts (advisors).	• Conducting 2.5h workshop with nine DevOps experts and three interviews with IT experts who were not able to participate in the workshop. • Gathering process-related and skill-related information about DevOps set-ups (Paper 4)
Expert interviews with DevOps thought leaders and industry experts	
Interviews with DevOps specialists and global key players and attending keynotes speeches held by them at practice conference and company visits.	Interviews with 7 global DevOps thought leaders to gain insights into the novelty of the DevOps concept and better understanding of practical implementation (Paper 7).

Attending practice conferences	
Active participation and presentation of research at practice-oriented conferences and seminars in the USA and Europe (e.g., DevOps Enterprise Summit)	• Exchange of experiences with practitioners and DevOps experts as well as knowledge sharing and presentation of prior research results. • Identification of participants for second case study (Papers 5-7).

Table 10. Summary of Research Methodologies (Own Depiction)

4. Research Studies

This section introduces the nine papers included in this dissertation. Part B1 includes accepted or published papers and Part B2 includes one paper currently under review for publication. Table 11 provides an overview of the eight publications.

No.	Authors	Title	Outlet and status	CON/JNL* (Ranking)
1	Wiedemann, Weeger	Developing Intellectual Capital within IT Teams: A Literature Review	European Conference on Information Systems, 2017 (accepted)	**CON** **(VHB: B)**
2	Wiedemann, Forsgren, Wiesche, Gewald, Krcmar	Research for Practice: The DevOps Phenomenon	Communications of the ACM, 2019 (accepted)	**JNL** **(VHB: B)**
3	Wiedemann, Wiesche, Gewald, Krcmar	Integrating DevOps within IT Organizations – Key Pattern of a Case Study	Projektmanagement und Vorgehensmodelle, 2018 (accepted)	**CON** **(VHB: N.A.)**
4	Wiedemann, Wiesche	Are You Ready for DevOps? Required Skill Set for DevOps Teams	European Conference on Information Systems, 2018 (accepted)	**CON** **(VHB: B)**
5	Wiedemann, Wiesche, Gewald, Krcmar	Implementing the Planning Process within DevOps Teams to Achieve Continuous Innovation	Hawaii International Conference on System Sciences, 2019 (accepted)	**CON** **(VHB: C)**
6	Wiedemann, Wiesche	How to Implement Clan Control in DevOps Teams	Americas Conference on Information Systems, 2018 (accepted)	**CON** **(VHB: D)**
7	Wiedemann, Wiesche, Thatcher, Gewald	A Control-Alignment Model for Product Orientation in DevOps Teams– A Multinational Case Study	International Conference on Information Systems, 2019, (accepted)	**CON** **(VHB: A)**
8	Wiedemann, Wiesche, Gewald, Krcmar	Understanding how DevOps Aligns Development and Operations: A Tripartite Model of Intra-IT Alignment	European Journal on Information Systems, 2020 (published online)	**JNL** **(VHB: A)**

* CON: Conference; JNL: Journal; VHB: German Academic Association for Business Research; N.A.: Not Available

Table 11. Overview of Accepted Publications of this Thesis (Own Depiction)

Table 12 presents the paper included in this dissertation that is currently under review for publication.

No.	Authors	Title	Outlet and Status	CON/JNL* (Ranking)
9	Wiedemann, Wiesche, Gewald, Krcmar	Combining Development and Operations: Toward a Control Model in Cross-Functional Teams	Information and Organization (1. revision)	**JNL** **(VHB: B)**
* CON: Conference; JNL: Journal; VHB: German Academic Association for Business Research; N.A.: Not Available				

Table 12. Overview of Publication in Review Process (Own Depiction)

PART B1

ACCEPTED PUBLICATIONS[1]

[1] The original publications are slightly modified, in terms of format, reference styles, the correction of spelling errors, and minor grammatical revisions. The original versions of the publications are included in the appendix.

1. Developing Intellectual Capital within Agile IT Teams

Title	**Developing Intellectual Capital within Agile IT Team: A Literature Review**
Authors	Wiedemann, Anna (anna.wiedemann@hs-neu-ulm.de) Weeger, Andy (andy.weeger@hs-neu-ulm.de) Neu-Ulm University of Applied Sciences Center for Research on Service Sciences Wileystraße 1 89231 Neu-Ulm Germany
Outlet	European Conference on Information Systems (ECIS), Guimarães, Portugal, 2017
Status	Accepted
Individual Contribution	Literature research, data collection, data analysis, identification of results, and manuscript writing

Table 13. Fact Sheet Publication P1 (Own Depiction)

Abstract

Since the Agile Manifesto was presented in 2001, more and more organizations move from traditional, plan-driven software development to agile approaches. This movement is motivated by the fact that environments are changing quickly and new requirements need fast implementation. We conducted a structured literature review to identify the current state of knowledge about agile IT teams and how they develop ambidextrous organizational learning to respond to rapid changes. We draw on the intellectual capital theory with the aim to explore key capabilities of agile IT teams of prior research. Afterwards, we synthesize the key capabilities considering intellectual capital. We derive intellectual capital configurations that enable IT teams to develop ambidextrous organizational learning. Furthermore, we identified technological oriented capabilities of infrastructure flexibility and architecture modularity for agile IT teams. Therefore, we built the concept of technological capital and arranged these capabilities. Thus, this study contributes to research by highlighting the characteristics that enable IT teams to be agile and thus helping companies to gain competitive advantage. Furthermore, we discuss possibilities how balance in ambidextrous organizational learning could be achieved. Additionally, we provide further research opportunities in this research stream.

1.1 Introduction

Following the publication of the Agile Manifesto in 2001, Agile Software Development (ASD) methodologies became very popular (Vial & Rivard, 2015) as they provide an effective method to react to rapidly changing customer requirements in challenging environments (Hummel, Rosenkranz, & Holten, 2013b). Hence, more and more organizations decided to move from plan-driven software development to agile approaches (Dybå & Dingsøyr, 2008; Tripp, Riemenschneider, & Thatcher, 2016; West, Grant, Gerush, & D'silva, 2010). Agility is defined as "*the ability to respond operationally and strategically to changes in the external environment"* (Fink & Neumann, 2007, p. 444). One of the most important differences between organizations that follow agile approaches and organizations that follow more traditional approaches is that they establish autonomous, self-organized teams, facilitate learning beyond knowledge silos and thus, facilitate and advance decisions-making within these teams (Coyle, Conboy, & Acton, 2015). For example, Amazon and Spotify replaced their traditional "silo" functions, which required enormous coordination efforts, with autonomous, cross-functional, product-centric teams that include a maximum of eight people (Advance IT Minnesota, 2016). This enables them to gain, share and implement knowledge, speed up decision making processes significantly and thus, to meet demand in rapid changing environments (Ramesh, Mohan, & Cao, 2012).

Developing innovative capabilities and realizing agile IT structures frequently enables companies to gain competitive advantages (Ramesh et al., 2012). However, companies need to be able to implement agile approaches to benefit from higher agility as reflected by shorter development cycles and strong collaboration within the IT team (Kude, Bick, Schmidt, & Heinzl, 2014). This seems to be difficult for rather old-fashioned companies. In contrast to modern organizations like Amazon and Spotify, these companies are more likely used to long software delivery lifecycles resulting from strict procedures and the realization of predefined plans (Coyle et al., 2015; Fitzgerald, 1998; Kude et al., 2014; Sommerville, 2009). In other words, these companies are quite effective in utilizing plans to refine and exploit existing knowledge, but have difficulties to bridge knowledge boundaries.

Prior research noted that organizational learning –the process of adoption and integration of new knowledge– is critical for realizing change (Kang & Snell, 2009). Since the publication of March (1991) a lot of investigations on organizational learning concentrate on the approaches of exploration and exploitation (Kang et al., 2007; Kang & Snell, 2009). Exploration is defined as the learning of new knowledge outside from companies knowledge stocks, whereas exploitation focus on improving and deepening existing companies knowledge domains (March, 1991). But firms are struggling to practice exploration and exploitation together. Hence, literature emphasizes the concept of ambidexterity within organizations to balance both, exploration and exploitation (Gupta, Smith, & Shalley, 2006) to foster organizational learning (March, 1991). Furthermore, ambidextrous learning is contingent on an organization's intellectual capital (IC). The several dimension of IC, human, social, and organizational capital play a significant role within the learning process (Kang & Snell, 2009).

In order to develop the ability to respond to changes in the external environment (i.e., the ability to be agile), IT teams need to perform ambidextrous learning that is balancing exploration of new, as well as exploitation of existing knowledge and related resources (Lee, Sambamurthy, Lim, & Wei, 2015). Although there is much literature on agile development teams, IS research is hitherto not clear on how IT teams could be enabled to develop this ability. To contribute to this gap, this paper sets out to synthesize prior research to illuminate how IT teams can be enabled to develop the ability of ambidextrous learning regarding the dimensions of IC. The following research questions guide this effort: *How can IT teams develop ambidextrous organizational learning capability to respond efficiently to environmental changes?*

We conducted a structured literature review to summarize the current state of knowledge and to approach this question. We analyze the findings of prior literature considering intellectual capital theory. More precisely, we analyze characteristics of agile teams that prior research has identified. Ultimately, we derive IC configurations that enable IT teams to develop ambidextrous organizational learning based on the approach of Kang and Snell (2009). Thus, this paper contributes to prior research by highlighting the characteristics that enable IT teams to be agile and thus, helping companies to gain competitive advantage.

The remainder of this paper is structured as follows. First, we discuss our theoretical understanding of agile IT teams, intellectual capital theory and ambidexterity in organizational learning. Then we outline our methods, present our findings and discuss the theoretical and practical implications of this research.

1.2 Theoretical Background

1.2.1 Agile IT Teams

Software development methodology is defined as an advocated approach for phases, rules, techniques, processes, documentation and training, employed for developing a software (Avison & Fitzgerald, 2003). In traditional development approaches, a sequential progression of phases is used and tasks are given to individual persons within separated organizational entities (e.g., business analyst unit, developer unit). Furthermore, this approach requires a huge amount of documentation, and much communication between the project participants is formalized through that documentation (Nerur, Mahapatra, & Mangalaraj, 2005).

Many of rather traditional structured IT organizations move to an agile, product-oriented structure of their entities to enable rapid software development. Normally, organizational entities are structured along several agile teams. An agile IT team is defined as a group of colleagues that work together with the aim to alter a running software or to build a new information system. Agile IT teams usually utilize agile methodologies such as Kanban, or Scrum, which create new possibilities to react fast with short development cycles and deliver software features iteratively to change customer or market requirements (Maruping et al., 2009).

Within agile IT teams all members should be able to perform all activities that are required to achieve the team objectives. Thus, members of agile teams should develop and advance a broad set of capabilities such as development, architecture knowledge, project management capabilities, and system administration (Tripp et al., 2016). Moreover, teams need to engage in continuous organizational learning processes to increase their performance. On the one hand, they are in the need to continually exploit existing knowledge, skills and capabilities (i.e., existing intellectual capital) and, on the other hand, to explore new knowledge such as that of their colleagues and customers (i.e., enhanced intellectual capital). Prior research shows that teams, which implement learning processes on group level early on, outperform other teams concerning both effectiveness and efficiency (Spohrer et al., 2012). Below we discuss the concepts of organizational learning and intellectual capital.

1.2.2 Intellectual Capital and Organizational Learning

Organizational learning consists of gaining, sharing, and integrating knowledge from outside the firm, as well as internal knowledge to improve on *"actions through better knowledge and understanding"* (Fiol & Lyles, 1985, p. 803). March (1991) provided a seminal of organizational learning that presents the trade-offs between exploration and exploitation, this model is widely used by existing literature (e.g. Gupta et al. (2006), Crossan, Lane, and White (1999)). Organizational learning appears on individual, group, and organizational level (Crossan et al., 1999; Kang & Snell, 2009) and involves the ability for exploitation and exploration. Exploitation focuses on a close, local, and deep search with repeating mechanisms to identify adequate solutions appropriate to firm's existing knowledge. Exploration is the result of a wide and generalized search to extend the knowledge fields of an organization into novel domains (Kang & Snell, 2009). The concept of ambidexterity refers to the ability to balance both exploration and exploitation. Consequently, literature on organizational ambidexterity focuses how these can be coordinated (Gupta et al., 2006).

As discussed above, agile, cross-functional IT teams are in the need for both exploration and exploitation within their teams (Ramesh et al., 2012). They need to exploit their skills and knowledge to develop the system in time and on budget. Organizations are able to encourage learning within subunits, since this could lead to the exploration of new possibilities (Fang, Lee, & Schilling, 2010). Moreover, they need to sense the environment, acquire new knowledge and skills to sense or anticipate, understand and respond to emerging changes (exploration) (Cao, Mohan, Xu, & Ramesh, 2009). Firms need an ambidextrous learning structure whereas they explore opportunities for new service development, as well as efficiently exploit existing products and services (Fang et al., 2010; O'Reilly & Tushman, 2008).

Prior research shows that organizational entities' abilities to develop or enhance organizational learning is closely tied to their knowledge inventory. In this regard, IC reflects this knowledge inventory, involving knowledge resources and capabilities of the networks the organization is embedded in. Consequently, IC refers to the combined knowledge that organizations can draw on to create a competitive advantage (Nahapiet & Ghoshal, 1998; Subramaniam & Youndt, 2005; Wang, Wang, & Liang, 2014). Concerning IT teams, it can be assumed that their ability to develop ambidextrous learning and thus to respond to changes in the external environment (i.e., the ability to be agile) is contingent on their IC.

Prior research has contributed a broad range of conceptualizations of IC. The perhaps most-widely accepted framework conceptualizes IC as consisting of three dimensions: human capital, social capital, and organizational capital (Subramaniam & Youndt, 2005). These dimensions reflect knowledge domains which are aggregated and distributed among the organization, its individuals, and on relationships between individuals (Kang & Snell, 2009; Subramaniam & Youndt, 2005). We assume that there are IC configurations that contribute to the ability of IT teams to develop both forms of organizational learning, exploration as well as exploitation and thus enable them to be agile. Kang and Snell (2009) provided a multilevel model of the dimensions of IC and discussed how they can be combined to foster ambidextrous organizational learning. Their research based on March (1991), they present two alternatives for each dimension of IC. Below we discuss the dimensions of IC and their alternatives.

Human capital refers to the combined knowledge of the individuals within a firm. This dimension encompasses the knowledge, capabilities as well as skills, and abilities which belong to and are used by the individuals (Subramaniam & Youndt, 2005). Concerning organizational learning, human capital can be either specialist and generalist in nature. Specialists possess deep, bounded, and embedded knowledge in special domains, whereas generalists are multi-skilled, can draw on a manifold repertoire of competencies and are able to apply these in different knowledge domains (Kang & Snell, 2009; Taylor & Greve, 2006). Specialized human capital may be accompanied by a functional bias which can decrease individual readiness and ability to vary and change new knowledge within their specified domain (Dougherty, 1992). Hence, specialists might rather focus on exploitation and ignore exploration (Kang & Snell, 2009). Generalists are more broadly oriented in various knowledge domains, generalist human capital has less functional bias. Furthermore, they can provide multiple solutions for a problem and apply, discover, and combine new knowledge. Subsequently, generalist are rather likely to focus on exploratory learning (Kang & Snell, 2009; Taylor & Greve, 2006).

Social capital reflects the knowledge that is available via relational networks. It is the pipeline for knowledge exchange and combination within the company. Firms can gain advantages through sharing knowledge and insights and/or mental models within working groups and social relationships (Kang & Snell, 2009; Karahanna & Preston, 2013; Nahapiet & Ghoshal, 1998). In this regard, Ghobadi and Mathiassen (2016) show that shared knowledge and communication within agile IT teams can positively influence project success, failure rates, and risk factors. Social capital comprises three dimensions; structural, relational, and cognitive (Nahapiet & Ghoshal, 1998; Wagner, Beimborn, & Weitzel, 2014). The structural dimension encompasses the overall scheme of network connections between individuals. This includes the communication settings between IT employees and other employees in form of meetings etc. (Wagner et al., 2014). The relational dimension of social capital contains the interpersonal exchange among individuals e.g., trust. This dimension focusses on the relationships between the individuals, which is essential for knowledge exchange and combination. The cognitive dimension implies to the degree of shared representations, understanding of individuals, and shared knowledge among individuals. Structural linking enhances relational and cognitive linkage and complete each other and social capital delivers the possibility for knowledge exchange (Kang & Snell, 2009; Karahanna & Preston, 2013; Nahapiet & Ghoshal, 1998). Prior research identified two generic social capital configurations: the cooperative and the entrepreneurial relational archetype of social capital. The cooperative relational archetype may facilitate exploitations as it involves strong and closely connected social networks characterized by strong trust and shared understanding (Kang et al., 2007). In contrast, the entrepreneurial relational archetype refers to more loosely coupled social networks. It enhances flexibility that is required to extent, gain, and absorb new knowledge and hence, it is expected to facilitate organizational exploratory learning.

Lastly, organizational capital refers to the institutional knowledge that is embedded in processes, structures, and systems. Organizational capital captures and integrates individual knowledge into organizational knowledge to be less dependent on individual employees (Grant, 1996; Kang & Snell, 2009). Based on prior literature, organizational capital can be categorized into two types: mechanistic and organic organizational capital (Kang & Snell, 2009). Mechanistic organizational capital refers to standardized structures and processes as well as tight rules. These are aimed at enhancing collaboration by determining enrooted patterns of interdependencies and providing detailed routines. Rules and standards facilitate integration of common opinions among colleagues. Thus, mechanistic organizational capital expected to facilitate exploitative organizational learning. Contrary, organic organizational capital refers to rather informal evolving rules, processes, and expectations about work results. Organic organizational capital enables continuous exploration, absorption and integration of novel knowledge (Kang & Snell, 2009) and is hence, related to explorative organizational learning.

Overall, prior research indicates that the ability to respond efficiently to environmental changes (i.e., being agile) requires ambidextrous organizational learning. Thus, we assume that characteristics of agile teams contribute to the ability of ambidextrous organizational learning. Ambidextrous organizational learning, in turn, seems to be contingent on the IC of an organization or organizational entity like an IT team respectively. Hence, we assume that there are IC configurations that facilitate ambidextrous organizational learning and help IT teams being agile.

1.3 Method

In general, literature reviews provide overviews on current states of research in distinct fields, synthesize existing knowledge and identify research gaps and unexplored research questions. As to that, reviewing literature is a necessary process in scholarship, which ensures that new research is connected to the existing body of knowledge within a research topic (Webster & Watson, 2002; Wolfswinkel et al., 2013).

Until now, no research was identified that synthesizes the characteristics that enable agile IT teams to develop ambidextrous organizational learning capability that are required to respond efficiently to environmental changes. Thus, we conducted a systematic literature review aimed at synthesizing the characteristics identified by prior research, to identify avenues for further research and to provide a framework that could guide such research.

The guidelines posed by Webster and Watson (2002) guided our efforts. Following their advice, we first defined criteria for inclusion or exclusion of articles. Our review explicitly focuses on characteristics that enable IT teams to develop ambidextrous organizational learning and hence contribute to their ability being agile. More precisely, we aim to come up with a classification of intellectual capital that enables agility. Hence, the emphasis of this review is on past investigations of the design of agile organizational entities related to IT (e.g., IT teams and IT functions). Particularly, we were looking for articles describing characteristics of intellectual capital of IT related organizational entities that enable ambidextrous organizational learning. Table 14 presents our criteria for excluding papers.

Reasons for exclusion
Research with strong focus on traditional IT teams/functions
Research that gives no insights into agile information systems development
Research with strong focus on organizational agility without delivering any insights into agile IT teams
Research which delivers no attributes/capabilities/components or insights for the design of agile IT

Table 14. Exclusion Criteria (Own Depiction)

Webster and Watson (2002) recommend to start the literature search with important journals and conference proceedings and to conduct a forward and backward search of selected articles. Since keywords like “agile”, “agility” and “IT or IS teams” resulted in too many irrelevant articles, which, in turn, increases the chance of missing articles that prove value to our research questions, we decided not to use a key-word based search strategy. Rather, we decided to manually scan the table of contents of premier IS research journals from North America and Europe (Schwartz & Russo, 2004; Wolfswinkel et al., 2013). Following the advice of Webster and Watson (2002), we identified relevant articles by scanning the title and abstract of each article within the selected journals and conference proceedings. If heading, key words and abstract suited our criteria, papers were included for further review. Following this strategy, 93 articles have been identified.

We included eight journals recommended by the Association for Information Systems (AIS) senior scholars’ group in 2011 (i.e., the senior scholars’ basket of eight) plus MISQ Executive. We included the latter one because it contains frequently cited practice-based publications in IS and is well respected in practice and academia. Furthermore, we included the past five years of the conference proceedings of the International Conference on Information Systems (ICIS) and the European Conference on Information Systems in our analysis (ECIS). By including the A- or B-rated conference proceedings, we intend to ensure that our literature review captures the latest research, which is not yet published in journals. We started our search in 2001, the year the Agile Manifesto was presented. We scanned all identified, published articles (93 articles) as described above and applied our exclusion criteria. Overall, 45 papers have been excluded. For the remaining 48 articles, we performed a backward research by checking the articles cited within the paper. Moreover, we used Web of Science and Google Scholar to perform a forward search to identify articles in other journals that cited these articles. These results have been analyzed using the same inclusion and exclusion criteria resulting in another 16 articles included in our analysis. In summary, we selected 64 papers (see Table 15).

Journal	Coverage	Hits	Included
MIS Quarterly	2001-2016	5	3
Journal of Strategic Information Systems	2001-2016	6	2
Journal of Management Information Systems	2001-2016	7	2
Journal of Information Technology	2001-2016	3	0
European Journal of Information Systems	2001-2016	11	6
Information Systems Journal	2001-2016	6	4
Information Systems Research	2001-2016	12	8
Journal of the AIS	2001-2016	9	5
MIS Quarterly Executive	2001-2016	5	3
International Conference on Information Systems	2011-2016	13	8
European Conference on Information Systems	2011-2016	16	7
Others Sources (forward/backward search)			16
Total		**93**	**64**

Table 15. Considered Sources and Number of Identified Articles (Own Depiction)

We coded the selected papers following the guidelines of Webster and Watson (2002). To mitigate the risk of coding bias, the research team has developed rigor rules that guided the coding and analysis process. One researcher went through the paper and coded passages covering information about the focus and theoretical base of the paper, considered characteristics of agile IT teams and the primary findings. During analysis, regular meetings were set up to discuss emergent findings, issues and divergent views, until agreement regarding the constructs reflecting intellectual capital dimensions has been achieved.

1.4 Findings

Several researchers investigated different settings of agile IT function or agile IT teams (Baumgart, Hummel, & Holten, 2015; Fink & Neumann, 2007; Tripp et al., 2016; Vial & Rivard, 2015). Analyzing and comparing the theoretical and empirical findings of these papers, we could identify eleven distinct characteristics related to intellectual capital, which help to foster ambidextrous organizational learning and thus, agility within organizational IT entities. The three dimensions of IC provide means for structuring the findings. Overall, literature confirms our assumption that IT teams need to employ both forms of organizational learning in order to be agile. Furthermore, we identified technology-oriented capabilities that could not be structured in one of the dimensions of IC. Consequently, we added a technological capital dimension. However, one could argue that technology could be closely related to the organizational dimension, we find much value in separating technological capital. Particularly since we are concerned with the IC of IT-related organizational entities.

Several papers within **human capital** related characteristics of agile IT teams like diversity, attitude, and requirements understanding. For example Lee and Xia (2010) highlighted, that agile IT teams that have a great diversity are more effective in responding to rapid environmental chances than homogenous teams. Literature emphasizes that the agile software teams' variety should fit the environment in which it is embedded, because diversity enhances the team's ability to react to changing environments.

Social capital encompasses the constructs shared knowledge, communication, as well as trust. Pries-Heje and Pries-Heje (2011) found that agile IT teams use effective communication ways and create trust. They conclude that this fosters social team capital. Social capital enhances IT teams understanding of business. Hence, this fosters knowledge exchange, enhances collaboration and accelerates strong ties within a network. Additionally, the team can achieve a better understanding of customer requirements through shared knowledge, trust, common language and communication (Wagner et al., 2014).

We identified governance, environment dynamism, and agile methodology use as constructs for agile IT teams that reflect characteristics of **organizational capital**. For example, Tiwana and Konsynski (2010) argue that enhancing agility is possible through IT governance decentralization. IT performance can increase through easing the exchange within an IT department (e.g., through decentralization).

Concerning **technological capital**, we found that prior research particularly investigates how infrastructure flexibility and architecture modularity enables agility. For example, new trends in technology faces infrastructure with great challenges, hence, a high degree of flexibility plays an essential role within agile IT teams. Prior studies associated IT infrastructure flexibility with fast business changes (Broadbent, Weill, & St. Clair, 1999; Salmela, Tapanainen, Baiyere, Hallanoro, & Galliers, 2015), and enhanced organizational responsiveness (Bhatt, Emdad, Roberts, & Grover, 2010; Salmela et al., 2015). These advantages are delivered by agile IT teams.

The following Table 16 presents the findings from the literature review structured per IC.

Human Capital

Characteristic	Definition of Construct	References
Diversity	Is defined as the extent to which the team members are different to each other, especially with focus on their skills, experience, and functional background that are necessary to solve problems, thinking in multiple ways and implementing solutions. Team diversity enhances the use of different talents.	(Kude et al., 2014; Lee & Xia, 2010; Spohrer et al., 2012; Tripp et al., 2016)
Attitude	Is defined as the perception, feelings, behaviors, and commitment of individuals towards the agile approach within their IT team.	(Cao et al., 2009; Ghobadi & Mathiassen, 2016; Schmidt, Kude, Heinzl, & Mithas, 2014; Spohrer, Kude, Schmidt, & Heinzl, 2013)
Requirements understanding	Is defined as the understanding of an agile IT team of customer's affordance for new features or necessary changes to enrich the software. Requirements have to be handled	(Cao, Mohan, Ramesh, & Sarkar, 2013; Fruhling & Vreede, 2006; Matook & Maruping, 2014; Overhage & Schlauderer, 2012;

	and implemented by the team and are likely to change during the development within an agile approach. A sequential collecting and refinement of requirements is necessary during the whole development.	Torkar, Minoves, & Garrigós, 2011; Vidgen & Wang, 2009)

Social Capital

Characteristic	Definition of Construct	References
Shared knowledge	Is defined as the team members shared, organized understanding of knowledge of essential key elements, roles and responsibilities, as well as tasks and skills of the environment of the team.	(Ghobadi & Mathiassen, 2016; Sambamurthy et al., 2003; Schmidt et al., 2014)
Communication	Is defined as the effective way to convey information between the team members. A clear communication and coordination process are dependent from the agile methodology use and is essential to avoid conflicts and redundant work. Agile team communication increases team members exchange frequency through short daily meeting, customer talks, and face-to-face communication.	(Botzenhardt, Meth, & Maedche, 2011; Harris et al., 2009; Hovorka & Larsen, 2006; Hummel, Rosenkranz, & Holten, 2013a; Li & Maedche, 2012; Scheerer & Kude, 2014; Schlauderer, Overhage, & Fehrenbach, 2015; Torkar et al., 2011; Tripp et al., 2016; Vial & Rivard, 2015; Wang, Conboy, & Pikkarainen, 2012)
Trust	Is defined as the ability to show widespread respect and to have a trustfulness relationship between two individuals or within the agile IT team.	(Baskerville & Pries-Heje, 2004; Baumgart et al., 2015; Dybå & Dingsøyr, 2008; Goh, Pan, & Zuo, 2013; Matook & Maruping, 2014)

Organizational Capital

Characteristic	Definition of Construct	References
Governance	Is defined as the degree of decentralization of decision-making authority within self-organized team-based structures of the IT department. Further, agile IT teams have a great autonomy in the decision-making power.	(Balaji, Ranganathan, & Coleman, 2011; Botzenhardt et al., 2011; Coyle et al., 2015; Krancher & Luther, 2015; Tiwana & Konsynski, 2010; Torkar et al., 2011)
Environment dynamism	Is defined as the agile IT team shared perceptions of practices, techniques, and procedures of an organization that are used to react on unpredictable and rapid changes of customers' preferences.	(Lee et al., 2015; Lowry & Wilson, 2016; Sarker, Munson, Sarker, & Chakraborty, 2009; Watts & Henderson, 2006)
Agile methodology use	Is defined as the extent to which the IT team uses agile software development methods as well as agile project	(Ågerfalk, Fitzgerald, & Slaughter, 2009; Balijepally, Mahapatra, Nerur, & Price, 2009; Baskerville

	management practices. Agile methodologies facilitate the rapid application development; have as little as possible documentation, active involvement of customers, and interaction and feedback processes as well as a strong focus on short cycle releases.	& Pries-Heje, 2004; Cao et al., 2009; Fitzgerald, Hartnett, & Conboy, 2006; Fruhling & Vreede, 2006; Mangalaraj et al., 2009; Maruping et al., 2009; Tripp et al., 2016; Wang et al., 2012)
	Technological Capital	
Characteristic	**Definition of Construct**	**References**
Infrastructure flexibility	Is defined as the ability of an IT team to react quickly and cost efficiently to customer demands. The infrastructure is flexible and compatible for emerging technologies and able to support running systems.	(Bush et al., 2010; Fink & Neumann, 2007; Goh et al., 2013; Rdiouat, Bahsani, Lakhdissi, & Semma, 2015; Salmela et al., 2015)
Modular architecture	Is defined as the degree to which an IT team can work within an architecture that disaggregated into autonomous subsystems to foster communication through standardized interfaces. This encompasses the loose coupling of applications, which means that changes in one system does not affect another system as well as competencies in emerging technologies.	(Choi, Nazareth, & Jain, 2010; Fink & Neumann, 2007; Tiwana & Konsynski, 2010; Yang, Antunes, & Tate, 2016)

Table 16. Findings of the Literature Review: Key Constructs of Agile IT Teams (Own Depiction)

The framework resulting from these findings is presented in Figure 3 below:

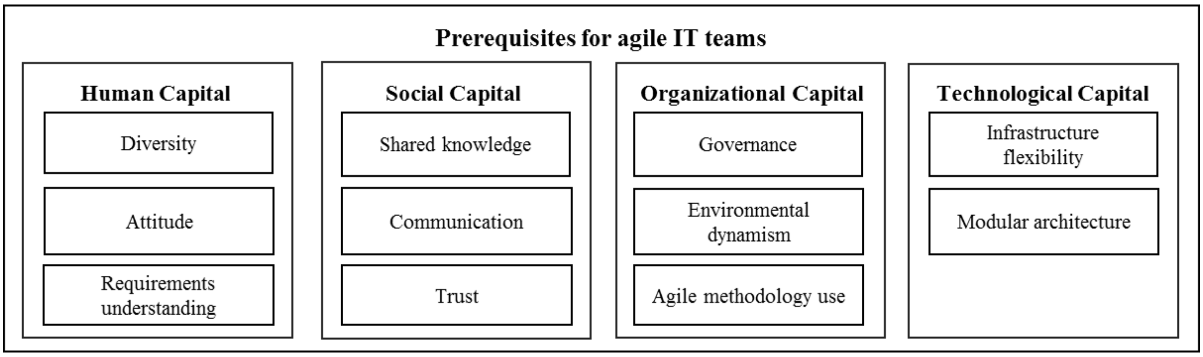


Figure 3. Findings: Four Dimensions of Key Constructs (Own Depiction)

The literature review largely confirmed our assumption. The results show the characteristics of agile IT teams contribute to ambidextrous organizational learning, which is dependent from IC. In the following, we present our findings for the different configurations of IC as well as for technological capital.

Generalist vs. specialist human capital: During the literature review, the research team identified different constructs, which reflect human capital characteristics. Individuals are important to uncover organizational boundaries and hence, foster the ability to receive and apply knowledge (Subramaniam & Youndt, 2005). Within agile IT teams, individual people with requisite special skills are one of the most important factors for a successful mode of operation. For example Chow and Cao (2008) explained that people based skills like high competency (diversity) are required to complete an agile project on time. This research shows that both aspects of human capital of organizational learning need to be given within agile IT teams. Furthermore, team members need an attitude, which is compliant with the mode of operation of the team. The team members need a commitment about the agile setting (Cao et al., 2009). Additionally, agile IT teams need general knowledge across a broad range of tasks that need to be delivered by the team (e.g., requirements understanding). IT teams should be able to represent all roles of their team that are necessary to complete their responsibilities (Tripp et al., 2016). This is a typical characteristic of exploration in organizational learning. Overall, our review indicates that agile IT teams need not just special knowledge in different topics and not only generalist understanding. Within an optimal setting, agile teams require a sufficient combination of both types of human capital. Thereby, the team should have a great diversity of skillsets to understand the customer's requirements, but also special knowledge to find the right solutions with the help of a motivated attitude to work with agile methods and react fast to moving environments.

Cooperative vs. entrepreneurial social capital: Prior research exhibits that success depends on the individual ability and willingness to work together to generate advantage for the company (Subramaniam & Youndt, 2005). Communication, shared knowledge, and trust play crucial roles for good collaboration within cross-functional teams. The challenge is to implement strong ties over all three dimensions of social capital within the team, namely structural, relational, and cognitive. Literature highlights that the individual knowledge of a particular team member should be shared among members of the agile group to learn from each other, and thus, foster group performance (Spohrer et al., 2012). Each team member should have close ties with the other team members (Pries-Heje & Pries-Heje, 2011). Agile IT teams need close cooperative ties between all members, which based on the individuals that have trust in each other to deliver exploitative learning. The entrepreneurial aspect of organizational learning is described by Sambamurthy et al. (2003) that agile IT teams have rather a strategic foresight, the ability to look ahead to changes within business environments. This describes rather weak ties to connection beyond the team. This emphasizes the entrepreneurial, exploration view of social capital. We recap that cooperative as well as entrepreneurial social capital should be implemented jointly within agile IT teams. To be ambidextrous, it is necessary to build strong networks within organizations to gain exploitation (Tushman & O'Reilly, 1996). Weak ties are important to achieve exploration in organizational learning.

Organic vs. mechanistic organizational capital: More and more teams are arranged decentralized within an organization. Consequently, new control, cultural and governance processes are required (Ågerfalk et al., 2009). Coyle et al. (2015) highlighted that a governance structure should have two-levels to be agile: a strategic level to enable communication and coordination among the company, and an individual level to foster decision making in a team-based structure. Within agile projects, it is necessary to make fast decisions to react on rapid environmental changes and thus, enhance communication. Literature shows that it is important for companies to integrate cross-functional teams within their IT department to foster organizational learning and gain better performance. Prior research depicted that teams outperform when a new product is developed by a responsible team on project level within the organization (Botzenhardt et al., 2011). Lee and Xia (2010) highlighted that agile development processes should be rather organic, dynamic, evolving instead of mechanistic, predefined and static. We summarize, that organizational capital of agile IT teams is on the one hand organic within the team level where the members must work autonomously and decisions have to be made quickly. This leads to exploratory organizational learning, because the teams search for new solutions and faster time to market delivery (Coyle et al., 2015). On the other hand, agile IT teams need predefined, mechanistic reporting lines for a suitable integration within the rest of an company (Botzenhardt et al., 2011). This contributes to exploitative learning, whereas the existing knowledge is consumed. Subsequently, agile IT teams need to balance forms of organizational learning to contribute to firm success.

Static vs. dynamic technological capital: Technological capital can be either static or dynamic. A strong strand of literature studied the trade-off between agility and stability, including issues related to exploitation and exploration (Gibson & Birkinshaw, 2004; March, 1991). Exploitation is associated with alignment, the degree how far the team fulfils customers' requirements and quality expectations under given technological conditions. IT projects are often based on static and well-defined time and cost schedules (Cao et al., 2013; Tiwana, 2010). Agile IT teams have to react fast, cost efficient and flexible on customer demands and hence, exploit existing infrastructure possibilities (Fink & Neumann, 2007). Hence, the static dimension of technological capital was identified. In contrast, exploration is connected to loosely coupled systems and technologies (He & Wong, 2004; Vinekar et al., 2006). Therefore, the dynamic dimension of technological capital was found. Agile teams work widely autonomously and should be able to decide which technology they use for their daily work. The teams need full flexibility to react quickly to changing environments and this may influence the group strategy (Li & Maedche, 2012). Highly modular systems enhance the ability to respond to changing markets (Tiwana & Konsynski, 2010). This contributes to explorative learning as the team is searching for new technologies to fulfil market demands. Additionally, system development projects must be aligned and controlled with client's needs and cost schedules. To summarize, agile IT team must explore the market for new technologies and ensure that they exploit the present technological conditions so that the customers are satisfied. Hence, both forms of technological capital are necessary within agile IT teams.

1.5 Discussion

Our findings extend existing knowledge through a literature-based overview of the settings of self-organized agile teams. Past literature primarily focused either on characteristics of agility, agile IT departments or connections of individual characteristics, for example autonomy or agile methodology use (Pries-Heje & Pries-Heje, 2011; Tripp et al., 2016). Hitherto, there is no research available that provides a broad overview regarding the characteristics of agile IT teams. Only a few researchers provide principles that go beyond single factors (Salmela et al., 2015; Tripp et al., 2016).

The results show that teams could have different settings of organizational learning. Our findings present the optimal setting of agile IT teams in case of ambidextrous organizational learning. Nevertheless, teams could also have other IC configurations in case of imbalance. Teams that have imbalance within organizational learning could lay too much weight on one type of human, social, and organizational capital. For example, if members have too much concentration on specialist knowledge (human capital) e.g., in form of adherence of deep knowledge about prior roles or job function, they will be not able to fulfil the other team roles. For agile IT teams it is a prerequisite to represent all roles of their team to achieve high performance (Tripp et al., 2016). To counteract to this phenomenon, agile IT teams could lay the focus on more generalist human capital with the aim to gain greater skill diversity (Chow & Cao, 2008). Thereby, the team might be able to reach more balance in organizational learning.

Furthermore, the various dimensions of IC could influence each other. The focus should lay on all dimensions of IC. Specialist human capital may influence cooperative social capital. For example, a barrier for agility within IT teams could be the missing attitude of individuals towards the agile approach. If individual team members are not willing to work in agile environment this might have an impact on imbalance of organizational learning. Team members with missing attitude may dislike sharing knowledge. Hence, we emphasize that teams should motivate their members to foster their attitude (organizational capital) towards agility. If team members have a high attitude towards agile approaches and methods they might be more able to build strong ties and e.g. effective knowledge sharing (social capital) (Ghobadi & Mathiassen, 2016). This might lead to a higher degree of ambidextrous organizational learning.

As we mentioned above, we identified technological capital as necessity to develop ambidextrous organizational learning within agile IT teams. We decided to determine this new dimension in order to raise awareness for this dimension because we delivered contributions in a research area of information systems. In addition, technological capital should be balanced with the dimensions of IC. For example, technological capital should be closely coordinated with organizational capital. The challenge is to exploit technological conditions to be aligned with clients cost schedules (Cao et al., 2013) as well as explore new technologies to react fast on unforeseen environmental changes (Tiwana, 2010). This might be influenced by the governance structure of IT teams and the IT department in that it is positioned. In order to achieve ambidextrous organizational capital, agile IT teams should have a great decision making autonomy but need also strong, predefined reporting lines to abide to organizations targets and orientation (Botzenhardt et al., 2011; Coyle et al., 2015). Thus, we emphasize to align the various dimensions of capital to achieve ambidextrous organizational learning.

1.5.1 Theoretical Implications and Further Research

Analyzing prior research and synthesizing characteristics of agile teams along IC dimensions that contribute to ambidextrous organizational learning capabilities helped to investigate the described gap. As to that, the framework and assumptions provide a starting point for further investigations considering the characteristics of agile IT teams, ambidextrous organizational learning and team performance. Further research may utilize the findings to relate different IC configurations to different levels of agility. Moreover, further research utilizing the framework would enhance our understanding on how IC of IT teams relate to their outcomes (e.g., alignment with customer needs, release frequency, etc.). Last not least, further research may investigate how the dimensions of intellectual capital related to ambidextrous organizational learning of IT teams reciprocally influence each other.

The present research highlighted that nearly all identified constructs could be ordered to IC. We identified a row of constructs within human, social, and organizational capital. Furthermore, we ordered the two constructs infrastructure flexibility (Salmela et al., 2015) and modular architecture (Cao et al., 2013; Tiwana & Konsynski, 2010) to the new dimension of technological capital. These aspects were identified as crucial factors within agile team settings. Additionally, we acknowledge two forms of technological capital, it can be static or dynamic. In addition, we found out that the combination of both forms is required to enhance the agility of IT teams.

We determine technological capital as separate dimension, because we like to show the importance of technological aspects within this area of information systems research. The given set of technology should be exploited as far as possible, but to achieve best and rapid solutions new technological possibilities should be explored. Hence, we complement existing research through characteristics of technological dimensions. Therefore, the presented framework could be enriched from further research, which investigates the correlation between human, social, organizational, and technological capital to enhance the theory.

1.5.2 Practical Implications and Further Research

We found out that IT departments who want to be agile need to balance IC with both forms of organizational learning. Kang and Snell (2009) suggested that the several dimensions of IC could be combined to enable a coexistence of exploitative and explorative organizational learning. We identified a row of constructs of agility that we ordered to the different dimensions of IC and to the characteristic of ambidextrous organizational learning. Forms of IC can be set into relation with each other and hence, foster internal alignment. Subsequently, we exhibit that self-organized agile teams must balance exploration and exploitation organizational learning of human, social, and organizational capital as well as technological capital.

We show that efficiency within these teams could be accelerated through the implementation of a set of skills and conditions if there is a clearing of exploration and exploitation. We highlighted that IC configurations that enable IT teams to develop ambidextrous organizational learning exist as a set of screws. Furthermore, we highlighted the necessity for ambidextrous organizational learning in case of technological capital. We depicted the static and dynamic dimensions. These screws can have different settings. We explained how these settings could be adjusted to gain higher balance of organizational learning.

Further research could explore possible configuration that might be applied in practice of the presented distinctions of organizational learning. For example, different combinations of constructs have different shapes in practice. The characteristic of human, social, organizational, and technological capital might vary in various organizations. The diverse settings of constructs of agile teams might lead to different outcomes. Additionally, further research might also consider alignment of the different dimensions within IT departments and their settings (agile, hybrid, or traditional).

1.5.3 Limitations

Though a structured literature search with the most relevant outlets in IS was performed, research in adjunct areas such as systems engineering might be also relevant to our research question. Subsequently, further research should include other research fields, and identify and analyze papers in outlets such as the Journal of Systems and Software. Second, the proposed synthesis reflects an extension of the few available typologies to characterize agile IT teams. However, until now, the concepts are solely grounded in prior research and need further theoretical amplification. Third, the concrete manifestation of the characteristics proposed above, their interrelations as well as effects need further elaboration – theoretically and empirically.

1.6 Conclusion

In this paper, we present the results of a structured literature review to gain understanding on how IT teams could develop ambidextrous organizational learning capability that enables them to respond efficiently to environmental changes? As to that, we argue that ambidextrous organizational learning is contingent on the IC of agile IT teams. Hence, we build up on the multilevel model which was delivered by Kang and Snell (2009). They described that firms can enhance ambidextrous learning if the several dimensions of IC are aligned.

We found much evidence for our assumption that agile IT teams need to balance exploration and exploitation in context of IC. Hitherto, we identified IC configurations that support such ambidextrous learning. The manifestations of human, social, organizational, and technological capital should be balanced within agile IT teams to make them efficient. With the help of prior literature, we identified eleven characteristics of agile IT teams that could be ordered to the IC dimensions and enable ambidextrous organizational learning. Our findings present evidences that the characteristics of agile IT teams contribute to ambidextrous learning, which is dependent on IC. The optimal setting of an agile team needs to combine both types of human, social, organizational, and technological capital. Ambidextrous organizational learning can be achieved, if agile IT teams foster the simultaneous appearance of the capabilities that are necessary for exploitation and exploration.

Furthermore, we derived a new dimension namely, technological capital and determined two specificities, static and dynamic. The static form of technological capital represents organizational learning in case of exploitation and the dynamic form exploration. We contribute to prior research by synthesizing the characteristics of IC that contributes to IT teams' ability being agile. Moreover, with the help of a literature review, we developed a research model that depicts these relationships and could guide further research. This research not only offers implication for further research, but also for practice, e.g., we show how imbalance within organizational learning could be adjusted.

2. The DevOps Phenomenon: An Executive Crash Course

Title	**The DevOps Phenomenon: An Executive Crash Course**
Authors	Wiedemann, Anna[1] (anna.wiedemann@hs-neu-ulm.de) Forsgren, Nicole[2] (nicolefv@gmail.com) Wiesche, Manuel[3] (wiesche@tum.de) Gewald, Heiko[1] (heiko.gewald@hs-neu-ulm.de) Krcmar, Helmut[3] (krcmar@tum.de) [1]Neu-Ulm University of Applied Sciences Center for Research on Service Sciences Wileystraße 1 89231 Neu-Ulm, Germany [2]DevOps Research and Assessment (DORA) San Francisco, USA [3]Technische Universität München, Chair for Information Systems Boltzmannstraße 3 85748 Garching, Germany
Outlet	Communications of the ACM, August 2019, Vol. 62. No. 8, pp. 44-49
Status	Accepted
Individual Contribution	Literature research, identification of results, and manuscript writing

Table 17. Fact Sheet Publication P2 (Own Depiction)

Abstract

A traditional software company releases its flagship product maybe every few years. Each release can include hundreds of new features and improvements. Because releases are infrequent, users can grow impatient waiting for each new release and are thankful when it finally arrives. Disappointment sets in, however, when bugs are found and features don't work as expected. Under great stress and with great turmoil, an emergency release is produced and put into production (hurried through the regular release process, often achieved by skipping tests), which has still more bugs, and the process repeats with more emergency releases, leading to more frustration, stress, and disappointment. Worse yet, new business opportunities are missed or ignored because of doubt, uncertainty, and distrust in the IT department's ability to deliver value.

2.1 Introduction

Isn't there a better way? Such practices are a thing of the past for companies that subscribe to the DevOps method of software development and delivery. New releases are frequent: often weekly or daily. Bugs are fixed rapidly. New business opportunities are sought with gusto and confidence. New features are released, revised, and improved with rapid iterations. In one case study, a company was able to provide a new software feature every 11 seconds (Sebastian et al., 2017). Which of these software teams would you rather be? Which of these companies will win during their industry's digital transformation? DevOps presents a strategic advantage for organizations when compared with traditional software development methods (often called phase-gate or waterfall) (Forsgren et al., 2018a). Leadership plays an important role during that transformation. In fact, Gartner predicts that CIOs who haven't transformed their teams' capabilities by 2020 will be displaced (Forsgren & Kersten, 2018).

2.2 Promises and Challenges for Digital Transformation

For organizations hoping to capture market share and deliver value faster (or even just deliver software more safely and securely), DevOps promises both speed and stability (Forsgren, 2018). It has developed as a prominent phenomenon of digital transformation in modern organizations that use software to deliver value to their customers in industries including banking, retail, and even manufacturing. DevOps combines activities of software development and delivery to enhance the speed of getting new software features to customers. This leads to higher customer satisfaction and profitability, which are important outcomes at the organization level. It also leads to important team-level outcomes such as better collaboration among different departments (such as developers, testers, and IT operations) and improved work-life balance. Executing a successful DevOps transformation isn't without its challenges. Organizations and software products vary in maturity and implementation, making transformation efforts difficult to design and deploy across teams and organizations. Most importantly, for DevOps to truly deliver value, it must include more than just tooling and automation—so simply purchasing and installing a solution isn't sufficient. As outlined here, DevOps includes culture, process, and technology. Indeed, success stories abound: Companies such as Kaiser Permanente, Capital One, Target, Starbucks, and ING have adopted DevOps methods, allowing them to deliver software for key applications in just seconds. DevOps enhances automation from applications to infrastructure provisioning. Continuous delivery supports automation and enables faster time to market and agile software development with fast feedback cycles. As the phenomenon is relatively new in practice, practitioners report on struggles with issues such as leadership and cultural transformation, implementing continuous delivery pipelines, and integrating a culture of collaboration in team settings (Nicole Forsgren et al., 2018a). Each of these areas would benefit from examination and guidance by formal research to test and augment the valuable experiences of those in industry.

2.3 A Move Away from Traditional Project Management

DevOps methodology is marked by a change in how software delivery is treated: moving from a discrete project to an ongoing product. The traditional way of delivering software (referred to previously as phase gate or waterfall) happens with the help of project management. In this paradigm, the project typically "ends" with the first major release of the new software or a set of new features delivered in a major incremental release. In general, the project team dissolves following a new release, and the responsibility for running the system is then "thrown over the wall" to operations. The operations team takes over responsibility for further changes and incident management. This leads to several problems:

- Developers are not responsible for running the system they built and therefore do not understand if trade-offs appear in creating and running the system, notably in the scalability and reliability of the software. This can lead to the same problems being perpetuated in future software releases, even if they are well understood by the operations team.
- The operations team is responsible for maintaining a highly reliable and stable system. Each new line of software deployment introduces change, and therefore leads to instability. This leads to a mismatch in incentives, where accepting new software from developers introduces risk (instability) and uncertainty (because less or no visibility into the development process gives them little insight into the software they are inheriting). Even if new components are of high quality, all new code adds complexity to the system and risk that the software was not developed with scalability and reliability in mind—key factors that operations must address and support in new software, as well as existing software.
- Stakeholders upstream (that is, earlier in the development process) from the production environment, including developers and the business, are not able to receive any feedback about performance until the first complete release. This generally takes place several months after project approval. Furthermore, not all features in software deliver value, and speeding up feedback to the development team and business allows for faster iteration to refine the software. This leads to wasted time and resources on features that don't ultimately deliver value. For example, research from Microsoft shows that only one-third of well-designed features delivered value to the customer (Kohavi et al., 2009).

In contrast, DevOps calls for a shift to product-based management. Practically, this means there is no more "end date" to projects, and teams instead deliver features— and therefore value— continuously. An important part of achieving this is integrating teams throughout the value stream, from development through operations; some organizations are even including business stakeholders. In this model, software is a product that is maintained as a product, with delivery and value metrics being tracked by the business continuously.

2.4 State-Of-The-Art Research and Practice on DevOps

One of the biggest challenges (and complaints!) in industry is the lack of a formal definition for DevOps. Many practitioners argue that this is intentional, because it allows teams and organizations to adopt a definition that works for them. In addition, they point out that having a formal definition of agile (coded in the Agile Manifesto) hasn't solved the problem of definition sprawl, and the resulting confusion around what is truly meant when an organization says it is "going agile" still plagues the industry. This lack of understanding can be challenging, so we present some common definitions for reference here.

- "DevOps is a software development and delivery methodology that provides increased speed and stability while delivering value to organizations (Forsgren, 2018)."
- "DevOps, whether in a situation that has operations engineers picking up development tasks or one where developers work in an operations realm, is an effort to move the two disciplines closer (Roche, 2013)."
- "DevOps, a compound of 'development' and 'operations,' is a software development and delivery approach designed for high velocity (Sebastian et al., 2017)."

And yet, these definitions focus on the outcome (value through speed and stability, via Forsgren) or the foundations of the discipline (bringing together development and operations, via Roche (2013) or Sebastian et al. (2017) If one wants to understand the components of DevOps methods, perhaps the most common presentation of these is summarized as CALMS: culture, automation, lean, measurement, and sharing (Forsgren & Humble, 2015; Humble & Molesky, 2011). Here is a brief definition of each component:

- Culture. Integration of mutual trust, willingness to learn and continuous improvement, constant flow of information, open-mindedness to changes and experimentation between developers and operations.
- Automation. Implementing deployment pipelines with a high level of automation (most notably continuous integration/continuous delivery) and comprehensive test automation.
- Lean. Applying lean principles such as minimization of work in progress, as well as shortening and amplification of feedback loops to identify and minimize value flow breaks to increase efficiency.
- Measurement. Monitoring the key system metrics such as business or transactions metrics and other key performance indicators.
- Sharing. Sharing knowledge in the organization and across organizational boundaries. Team members should learn from each other's experiences and proactively communicate.

2.5 An Interpretation of Calms

This article uses the five core elements of CALMS as a framework.

2.5.1 Culture

Working as part of a DevOps team requires a culture of collaboration within a cross-functional team setting. In its ideal state, DevOps uses a so-called cross-functional team, which means that groups made up of developers, testers, quality assurance professionals, and IT operations engineers all work together to develop and deliver software. In this way, they are familiar with each other’s work and challenges (a common phrase to describe this is “to have empathy”), which helps them create and maintain better software. For example, because they see the challenges in maintaining scalable and reliable infrastructure encountered by operations professionals, developers write more scalable and reliable code in collaboration with operations staff.

2.5.2 Automation

The next principle, automation, requires a suite of DevOps tools (Ebert, Gallardo, Hernantes, & Serrano, 2016; Humble, 2018). Following are a few examples of available automation tooling:

2.5.2.1 Build

- Build. Tools for the software development and service life cycle, including compiling code, managing dependencies, creating documentation, conducting tests, and deployments of an application in different stages.
- Continuous integration. Constant testing of new software components.

2.5.2.2 Release

- Release automation. Packaging and deploying a software component from development across all environments to production.
- Version control. Managing changes to the program and collecting other information. Version control is a component of configuration management.
- Test automation. Using software to execute tests and repeating activities.

2.5.2.3 Deployment

- Configuration management. Reducing production provisioning and configuration maintenance with the help of reproducing the production system on development stages.
- Continuous delivery. A combination of principles, practices, and patterns designed to make uninterrupted deployments

2.5.2.4 Operations

- Logging. Traces are essential for application management; everything must be logged.
- Monitoring. Identification of infrastructure problems before the customers notice them. Monitoring storage, network traffic, memory consumption, etc.

Continuous integration and delivery reduce the cost and risk of releasing software. They do this by combining automation and good practices to consistently, reliably, and repeatably perform work (such as tests and builds) that enables fast feedback and builds quality in during the software-delivery process. This helps teams deliver features faster and more reliably and, in turn, achieve faster value delivery for the organization. The highest performing teams are able to deploy 46 times more frequently with 2,555 times shorter lead times than low performers. Failure rates are seven times less, and they are able to recover 2,604 times faster than their lower performing peers (Nicole Forsgren et al., 2018a).

2.5.3 Lean

"Building quality in"—referenced in the previous paragraph—is a key tenet in DevOps, and a principle also found in lean. Applied to DevOps, this means teams look for opportunities to remove waste, leverage feedback loops, and optimize automation.

Let's look at an example. DevOps teams differ in size and product responsibility. In some models, a single team conducts all software development and delivery activities, including development, testing, delivery, and maintenance. They are ultimately responsible for the complete software-delivery life cycle of the software products (and may be responsible for more than one product), delivering value to the business. Lean processes allow for quick iterations and feedback throughout the development and delivery process to improve quality and build faster and more reliable systems. Examples include working in small batches to enable fast flow through the development pipeline, limiting work in process, fixing errors as they are discovered vs. at the end, and "shifting left" on security input. These practices may sound familiar; similar practices have driven quality and value in manufacturing. (For a great story about lean manufacturing, check out episode 561 of This American Life,11 which discusses NUMMI (New United Motor Manufacturing Incorporated), the joint venture between Toyota and GM in Fremont, CA.)

2.5.4 Measurement

Measurement is another core aspect of DevOps. The ability to monitor and observe systems is important, because software development and delivery are essentially dealing with an invisible inventory that interacts in complex ways that cannot be observed. (This is in contrast to traditional physical manufacturing systems such as an automobile assembly line, described in the NUMMI case.) Through effective monitoring, teams are able to track, watch, measure, and debug their systems throughout the software-delivery life cycle. It should be noted that metrics are also a tool for quality assurance, and measurements from several sources should be leveraged (Forsgren & Kersten, 2018).

2.5.5 Sharing

Sharing of knowledge and information enables successful DevOps teams and helps amplify their success. By sharing practices—both successes and failures—within teams, across the organization, and across the industry, teams benefit from the learning of others and improve faster. While others have pointed out that sharing is possible in any domain and any methodology, DevOps has adopted this as a cultural norm, and many in the industry report that the field is much more collaborative than their prior work in tech.

Internal collaboration may include work shadowing or job swapping: developers are involved in operations and maintenance activities (for example, developers may even “take the pager”), and operations engineers rotate in to development and test roles, learning essential components of design and test work. In many cases, all cross-functional team members participate in the same meetings, which gives them shared context. Cross-industry sharing often takes places at conferences, with dozens of DevOpsDays and other community-organized events sprouting up around the world.

The application of these principles leads to better outcomes: for individuals (seen in reduced burnout and greater job satisfaction), for teams (seen in better software delivery outcomes and better team cultures), and for organizations (seen in improved performance in measures such as profitability, productivity, customer satisfaction, and efficiency (Nicole Forsgren et al., 2018a; Wiedemann, 2017). Although DevOps has been an important movement in industry for more than a decade, it hasn’t received much attention from the academic community until recently. And while CALMS principles are not always referred to using these terms, they do appear in existing research (e.g., Fitzgerald and Stol (2017)).

2.5.6 Research Summaries

The core insights of some prior DevOps-related research follow:

- Fitzgerald and Stol (2017) The Journal of Systems and Software presents a continuous software-engineering pipeline and a research agenda for different continuous processes including DevOps and BizDev (business strategy and development).
- Forsgren, Sabherwal, and Durcikova (2018b) European Journal of Information Systems highlights the roles adopted when sharing and sourcing knowledge in a system management context, and identifies strategies for optimizing outcomes.
- Forsgren, Durcikova, Clay, and Wang (2016) Communications of the AIS identifies the most important system and information characteristics of tools for users who maintain systems.
- Dennis, Samuel, and McNamara (2014) Journal of Information Technology highlights that maintenance effort should be considered during the design phase and calls this DFM (design for maintenance). The authors present insights into how the links among documents should affect both the maintenance effort and use.
- Sharma and Rai (2015) European Journal of Information Systems investigates how an organization’s computer-aided software engineering adoption decision is influenced by individual factors of IS leaders and technological factors.
- Shaft and Vessey (2006) Journal of Management Information Systems aims to examine software maintenance as interlinking comprehension and modification; the relationship between these two factors is moderated by cognitive fit.

2.5.7 Conferences

- Trigg and Bødker (1994) ACM Conference on Computer Supported Cooperative Work examines how people tailor their shared PC environment and presents an understanding of how software development tailoring can be helpful in designing systems that better fit the demands.
- Lwakatare et al. (2016) Hawaii International Conference on System Sciences investigates key challenges of DevOps adoption in embedded system domains.
- Wiedemann and Wiesche (2018a) European Conference on Information Systems presents an ideal skill set that DevOps team members should adopt to manage the software delivery life cycle.

2.6 Practical Challenges and Opportunities

Implementing DevOps presents several challenges.

First, technology—and the resulting organizational transformation—are difficult, but strong leadership can help. As many practitioners note, managing and enabling the cultural changes inside an organization can be a more difficult and important challenge than implementing the technical changes. Good leadership is vital. One approach studied in several contexts, including DevOps, is transformational leadership; this style uses five dimensions (vision, intellectual stimulation, inspirational communication, supportive leadership, and personal recognition) to inspire and guide teams.

Second, DevOps requires a custom solution for each organization. Each context is unique, and a prescriptive approach to DevOps implementation and adoption is unlikely to be successful. Teams and organizations pose a unique set of challenges and cultural norms. Each one should adopt and adapt its own approach to achieve DevOps success. The adaptation of DevOps should include development of not only technical, but also cultural, process, and measurement practices. The work of creating a unique and seemingly ad hoc technology transformation journey is difficult and may be daunting, so many organizations look for step-by-step guides. However, these are not likely to provide solutions (beyond basic advice such as “automate your toolchain”) and are usually offered by those trying to sell you something.

Third, each DevOps solution should encompass a holistic view, consisting of automation (including tools and architecture), process, and culture. In many traditional approaches, specialist knowledge in one area (e.g., development) is leveraged to accomplish a task before passing it off to another group. In DevOps, a move from this high specialization to include a broad understanding of more areas is necessary. (Some call this T-shaped knowledge, with the top part of the “T” representing broad knowledge, while the stem of the T represents deep understanding in one area of expertise.)

This allows people to understand how their work will affect and interact with more areas of the technical stack; this often requires significant additional learning and responsibilities in the transition to DevOps. Organizations should provide training and education, and not just expect technologists to augment their learning independently. Note that while some technologists consider this expansion of responsibilities and knowledge exciting, others may push back, especially those who are just a few years from retirement and comfortable in their work roles, or who see sharing information about their roles as a risk to job security. Organizations will need to consider these training and cultural challenges in particular and respond accordingly. (As already noted, DevOps is about much more than just technology!)

2.7 Conclusion

DevOps is about providing guidelines for faster time to market of new software features and achieving a higher level of stability. Implementing cross-functional, product-oriented teams helps bridge the gaps between software development and operations. By ensuring their transformations include all of the principles outlined in CALMS (culture, automation, lean, metrics, and sharing), teams can achieve superior performance and deliver value to their organizations. DevOps is often challenging, but stories from across the industry show that many organizations have already overcome the early hurdles and plan to continue their progress, citing the value to their organizations and the benefits to their engineers.

3. Integrating DevOps within IT Organizations–Key Pattern of a Case Study

Title	**Integrating DevOps within IT Organizations–Key Pattern of a Case Study**
Authors	Wiedemann, Anna[1] (anna.wiedemann@hs-neu-ulm.de) Wiesche, Manuel[2] (wiesche@tum.de) Gewald, Heiko[1] (heiko.gewald@hs-neu-ulm.de) Krcmar, Helmut[2] (krcmar@tum.de) [1]Neu-Ulm University of Applied Sciences Center for Research on Service Sciences Wileystraße 1 89231 Neu-Ulm, Germany [2]Technische Universität München Chair for Information Systems Boltzmannstraße 3 85748 Garching, Germany
Outlet	Projektmanagement und Vorgehensmodelle, Düsseldorf, Germany, 2018
Status	Accepted
Individual Contribution	Literature research, data collection, data analysis, identification of results, and manuscript writing

Table 18. Fact Sheet Publication P3 (Own Depiction)

Abstract

Team-oriented, service-centric approaches for software delivery processes are becoming more and more popular. One approach that has appeared on the scene in the last decade is called DevOps. To date, little is known about how cross-functional product-oriented teams can be integrated within traditional companies. Hence, we decided to contact organizations that have already started implementing DevOps-specific processes. We talked to 34 people from 10 organizations and derived new insights into the area of DevOps. We show that there are different patterns by which DevOps can be integrated within companies—e.g., with the help of platform-oriented teams. Further, we find that teams are organized in different setups and use extended workbenches for collaboration. Our case study highlights that for successful DevOps implementation, it is important to convince every stakeholder and to work with agile coaches who can foster awareness for the necessity of DevOps.

3.1 Introduction

Digital transformation has led to new challenges for IT functions. A lot of companies are searching for new insights to manage their IT departments for meeting new customer requirements, because their working mode is rather reactive. Many incumbent companies are at an early stage of digital transformation and are under pressure to adapt their capabilities to develop digital strategies (Pflügler, Wiesche, & Krcmar, 2018; Sebastian et al., 2017).

IT departments have to work with new technologies, foster their business understanding, and focus on the delivery of new software features to customers (Przybilla, Wiesche, & Krcmar, 2018). Therefore, in recent times, service-centric IT team approaches are appearing and companies have started to implement the so-called DevOps IT teams. DevOps is a portmanteau of the words development and operations (Bharadwaj, El Sawy, Pavlou, & Venkatraman, 2013; Lwakatare et al., 2016). Literature highlights that there is no common definition of DevOps (Fitzgerald & Stol, 2017). We define DevOps as a concept for integrating the tasks, knowledge, and skills pertaining to the planning, building, and running of activities in a single cross-functional team that is responsible for the combined development and operational tasks of one or more software service products. To quickly deliver new software features and innovations, ensure the quick handling of problems, and integrate maintenance activities, IT departments should integrate cross-functional teams rather than silo-organized IT units (Fitzgerald & Stol, 2017). Aligning development and operational tasks within one team can lead to higher IT and organizational performance. High-performing DevOps IT functions are able to deploy 46 times more frequently than low-performing ones (Brown, Forsgren, Humble, Kersten, & Kim, 2016). DevOps is a rising trend and publications predict that it will become a competitive necessity. More and more IT organizations are deciding to implement the DevOps concept (Forsgren & Kersten, 2018; Ross et al., 2016). According to a Gartner survey from 2016, about 25% of 2,000 global IT companies will work with the DevOps concept and tools in the DevOps toolchain in the future (Rivera & van der Meulen, 2015). Hence, DevOps is being recognized as an important topic in the area of software development processes and process models. DevOps is a rising trend (Wiedemann & Wiesche, 2018a). Hence, we have decided to conduct a case study with companies that are already working with DevOps principles. Taking into account the fact that the companies implement DevOps in different ways, we investigate the following research question: What are the key patterns of DevOps teams?

3.2 The DevOps Concept

Patrick Debois is considered as the person who branded DevOps. He held the first DevOpsDays conference in Ghent, Belgium in October 2009 (Devopsdays, 2009; Reed, 2014). As mentioned above, DevOps combines two words—development and operations. Key activities of DevOps are automated development, deployment, and monitoring of infrastructure within one cross-functional team. Organizations start using service-centric teams, integrating different roles and responsibilities, and breaking down historically grown silo departments with a high level of specialist knowledge. With the DevOps movement, a cultural shift toward better collaboration between developers, operations people, and quality assurance is necessary. The DevOps concept is helpful for delivering faster value to customers through timely provision of new software components, reducing problems through miscommunication, and enhancement of problem resolutions (Ebert et al., 2016).

Prominent examples like Amazon and Google have already adopted principles of the DevOps concept and now have cycle times of new software components in seconds (Sebastian et al., 2017). The DevOps approach enables the scaling of agility to the entire IT organization. The goal of DevOps is to enhance collaboration, automation, virtualization, and tools to bridge the activities of software development and operation (Gotel & Leip, 2007; Humble & Molesky, 2011; Lwakatare et al., 2016). Through DevOps, solutions are delivered to avoid interruptions between different stages of the software delivery process (Fitzgerald & Stol, 2014). The software development lifecycle consists of planning, development, and operation tasks. DevOps helps companies to integrate the necessary speed and flexibility to deliver constant and rapid development and implementation of digital innovation. Hence, risks linked with software releases can be reduced, and feedback of new software features is received faster. In addition, agile methods can be used to manage the software development part of the team (Lwakatare et al., 2016). In agile development environments, there are different "continuous" activities, the best-known of which is continuous integration (CI). CI is defined as a process that is provoked automatically and encompasses inter-connected stages like acceptance test, code validation, compliance checks, and release package development (Fitzgerald & Stol, 2014). CI disables interruptions between the development and deployment stages of software delivery. It is very significant for the DevOps phenomenon because good collaboration between development and operation is needed (Fitzgerald & Stol, 2014). The benefits of CI are improvements in communication, higher frequency in releases, and an enhancement in the productivity of developers (Fitzgerald & Stol, 2014; Kim, Park, Yun, & Lee, 2008; Ståhl & Bosch, 2014). CI can lead to other continuous activities like continuous deployment (CD) and delivery(Fitzgerald & Stol, 2017). DevOps could help to enable these processes.

DevOps complements and extends CI and releases agile software development processes by bridging new software features quickly into production and value delivery to the customer. DevOps teams can use agile methods to manage collaboration and work within their deployment environments. Companies use DevOps for better collaboration and monitoring in order to enhance the continuous delivery of new software features and consequently foster innovation (Lwakatare et al., 2016).

3.3 An Investigation of DevOps Teams

As mentioned above, there are some prominent examples of companies already working with DevOps approaches. However, little is known about how the DevOps concept is adopted in existing companies. Hence, we decided to contact organizations that have already adapted the DevOps approach. Therefore, we searched through the internet and social business networks (e.g., Xing), and contacted people in various companies responsible for DevOps. Thus, we were able to find 10 participants for our case study. We held two or more interviews per case. After conducting the interviews, we identified different characteristics that were adopted by each case, which we grouped to four different categories:

- **Organization of IT function:** This describes how the IT function is organized and how the DevOps approach implementation begins.
- **Implementation core and product:** This describes the DevOps team setting and service.
- **CI/CD:** This describes the degree of CI/CD pipeline implementation and the deployment rates.
- **Extended workbench:** This describes how the DevOps team is organized (i.e., with greater development or operations background) and the degree of work task organization within the DevOps team.

We present the characteristics of the DevOps setup in the similar cases in Table 19 and 20.

3.3.1 Data Collection

To answer our research question, an exploratory multiple-case-study approach is adopted for a number of reasons. The case study approach is defined as “an empirical inquiry that investigates a contemporary phenomenon in depth and within its real-life context” (Yin, 2014, p. 18). DevOps is quite unexplored and case studies are helpful to examine new phenomena from various perspectives (Yin, 2014). The advantage of case studies is that they can zoom in on real-life situations and test or develop theoretical views in relation to phenomena, as they unfold in practice (Flyvbjerg, 2006).

Case	Organization of IT function	Implementation of core and product
Case 1 (Media)	Newly founded company with an IT function with high DevOps orientation.	The team is responsible for a data service for website delivery (internal service).
Case 2 (Service)	Transformation toward DevOps for several years.	The team is responsible for a product of their website (e.g. insurance service).
Case 3 (Home)	Ad hoc decisions to reorganize the IT function with DevOps teams.	The team is responsible for an activity of the online shop (e.g. shopping basket).
Case 4 (Pet Industry)	Strategic decision to reorganize the IT functions with the help of DevOps teams.	The team is responsible for an activity of the online shop (e.g. product rating).
Case 5 (Specialized Store)	Strategic decision to integrate DevOps teams for managing their online shop.	The team is responsible for an activity of the online shop (e.g. check-out).
Case 6 (Retail 1)	New company with the idea to integrate high DevOps orientation.	The team is responsible for an app development and support (e.g. delivery service).
Case 7 (Insurance)	Integration of the DevOps approach within an existing IT function.	The team is responsible for an internal delivery platform and offers self-service to other teams (internal service).

Case 8 (Bank)	Integration of the DevOps approach within an existing IT function.	The team is responsible for running and facilitating a securities management system (internal service).
Case 9 (Retail-non-food)	New company with the idea to integrate high DevOps orientation.	The team is responsible for building and running an internal platform, which is the substructure of the online shop of the company (internal service).
Case 10 (Retail 2)	Integration of the DevOps approach within an existing IT function.	The team is responsible for building and running an internal platform, which is the substructure of the online shop of the company (internal service).

Table 19: Findings of the Case Study Research (1/2) (Own Depiction)

Case	CI/CD	Extended workbench
Case 1 (Media)	CD is integrated. The team is able to deploy around every four days.	The team consist of 12 software engineers located in Germany and with a nearshoring partner in East Europe and shares development and operations work.
Case 2 (Service)	Frequent releases but no optimal CI/CD.	The team consists of eight software developers and is located in Germany. The team lead is responsible for the operations parts.
Case 3 (Home)	CD is integrated. The team is able to deploy several times a day.	The team consists of five software engineers. It is located in Germany and shares development and operations work.
Case 4 (Pet Industry)	No optimal CI/CD. The team is able to deploy every two weeks.	The team consist of four developers and is supported by a quality assurance engineer. The team is located in Germany and shares development and operations work.
Case 5 (Specialized Store)	No optimal CI/CD. The team is able to deploy every two weeks.	The team consists of five software developers and operations people. It is located in Germany and shares development and operations work.
Case 6 (Retail 1)	CD is integrated. The team is able to deploy every two weeks.	The team consists of four software developers and is located in Germany. The infrastructure team is a separated unit and is responsible for operations during the day.
Case 7 (Insurance)	CD is integrated. The team is able to deploy several times a day.	The team consists of 15 software engineers allocated in Germany and Asia and shares development and operations work.
Case 8 (Bank)	No optimal CI/CD. The team is able to deploy once a week.	The team consists of 15 software operations people and is located in Germany. The developers are a separate unit.

Case 9 (Retail-non-food)	CD is integrated. The team is able to deploy several times a day.	The team consists of six software operations people and is located in Germany. The developers are in a separated unit.
Case 10 (Retail 2)	No optimal CI/CD. The team is able to deploy every two weeks.	The team consist of three software developers and is located in Germany and with a nearshoring partner in Bulgaria. The infrastructure is managed by another subsidiary company of the group.

Table 20. Findings of the Case Study Research (2/2) (Own Depiction)

3.3.2 Data Analysis

We applied a multiple-case-study design that enables cross-case pattern search. This method helps to examine processes and patterns over several cases to understand how similar or contrasting the results are (Miles & Huberman, 1994; Yin, 2014). The data were analyzed with the help of “within-case analysis” as well as “cross-case-analysis”. In the within-case analysis, each team was seen as a stand-alone entity and analyzed (Yin, 2014).The analysis process was conducted through the lens of the key pattern. The focus was on cross-case analysis, to compare the cases and identify the patterns obtained for each case. In the present research, the coding approach emphasized by Miles and Huberman (1994) has been applied. Afterwards, in vivo coding was applied to examine each case and emergent topics or explanations. Additionally, it helps to achieve more familiarity with each case and fosters cross-case analysis (Miles & Huberman, 1994; Yin, 2014)

3.4 Key Findings and Discussion of the Case Study

In the following section, we present several insights achieved through our case study, as well as the key pattern identified in our research.

3.4.1 Insights from the DevOps Teams

To summarize, we talked with 34 persons from the aforementioned companies. All of them offer different insights into the DevOps setting. During our investigation, we recognized that there are different patterns across the several cases. Nevertheless, we found some similarities as well. Table 21 presents an illustrative example of the key findings that we identified in the different IT functions. The table presents two patterns generated with the help of our case study.

Cases 5, 6, 7, 9, and 10 have integrated a so-called platform team, which they have organized with the help of the DevOps approach. That means that an internal platform is used as the basic infrastructure—as the foundation for the online shop of the organizations. The infrastructure teams are often organized with the help of operations people who have high interest in development tasks as well. For example, they develop their own software program and write playbooks, mostly using Linux as the operating system. Hence, for them, it is a key requirement that new team members should have knowledge in Linux.

DevOps platform teams usually have operations background and undertake on-call duty outside normal business hours or developers with experience in system administration. The development teams are responsible for their services during the normal business hours. For a couple of reasons such as costs, companies may decide not to integrate on-call duty in the development teams. Very often, a lot of deployments are made during the day.

Companies start implementing some rules and regulations for deployments. For example, one case mentions that new software components are not allowed to be deployed into the production environment on Fridays, because they want that the system to be running stably during the weekend. Another case mentions that if a developer deploys new software components after the regular deployment times, the person has to be available for the DevOps platform team in the case of failure messages. The overall aim of the platform DevOps teams is to integrate self-services for the development teams. This means that developers are able to help themselves with the particular services they need (e.g., databases or firewalls). The DevOps platform team serves these services to the development teams with the help of application program interfaces (APIs).

Pattern 1	Pattern 2
DevOps teams have an internal platform orientation with self-service opportunity for development teams.	The IT department mainly consists of DevOps teams that are working with e.g. micro-service architecture.
Most team members have operations background and have started working with development tasks.	Most team members have software development tasks and have to learn operations tasks.
The team is mainly responsible for providing infrastructure to the development teams and supporting them.	The team is mainly responsible for service and infrastructure. If superordinate problems appear, they are supported by a support unit.

Table 21. Pattern of DevOps Teams (Own Depiction)

The other cases that we investigate present insights into another DevOps setup. Cases 1, 3, 4, and (partly) 2 claim that they have integrated the development and operations tasks within one team. The CTO of Case 3 mentions that it is important for the development team to be fully responsible for its service. Most of the cases are responsible for one service of an online shop (e.g., shopping basket, check-out processes). If problems appear, they have to manage it as well. The DevOps team has a high decision-making freedom regarding new technology. Furthermore, it has to guarantee that skills and knowledge necessary for managing the service are integrated within the team. For example, Case 2 mentions that it is not possible for everyone to know everything. The team should be organized so as to ensure that the service is working, even if team members are on holiday or unwell. Hence, team members should be able to stand in for each other.

IT organizations and companies organize meetings, tech-talks, and other presentations to share knowledge within and outside the company. Sharing knowledge is a key factor in the DevOps movement (Bang, Chung, Choh, & Dupuis, 2013). These knowledge-sharing activities are helpful because the teams learn from each other. Additionally, the teams are supported by agile coaches or disciplinarians. These people manage the teams and help them with decisions —for example, deciding whether new technologies are really necessary if the technology already exists within the company with the help of another tool. Figure 4 visualizes two examples of the different DevOps setups observed in several cases.

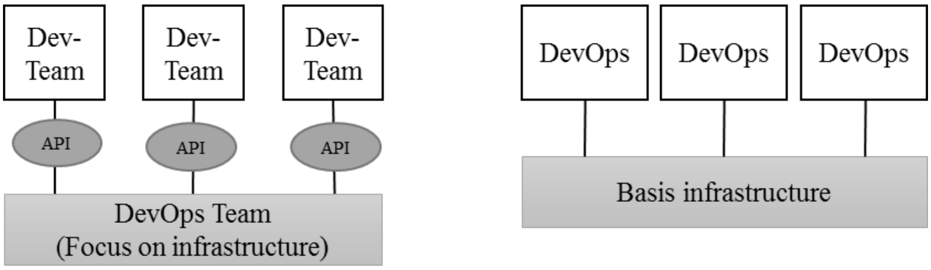


Figure 4. Organizational Structure of DevOps (Own Depiction)

Furthermore, our findings show that the integration of DevOps presents challenges for the existing IT function and organizational structure. We recognize that the IT functions—whether completely reorganized with DevOps teams or organized as bimodal IT functions—are traditional silo units and DevOps teams coexisting.

3.4.2 Starting Points for DevOps

Another finding of our case study is that a lot of tradition-oriented companies are searching for ways to implement DevOps within their company. The study participants mention different starting point for the implementation of DevOps within new or existing IT functions. We summarize the following four different starting points for DevOps implementations:

- Spin-offs/subsidiary companies
- Greenfield projects
- Slow accession process
- Ad-hoc adjustment

Our findings show that different starting points for a successful implementation are possible. These opportunities should be considered by organizations trying to implemented DevOps.

3.4.3 Movement Towards Product Orientation

A major finding of our study regarding the organization of DevOps teams is that they do not usually work in a project setup. They are organized in the so-called product setups. IS projects usually have pre-defined project aims such as project delivery on-time and within-budget (Kirsch, 2004). Our findings suggest that there is a shift toward product-orientation—this means that there is no regular starting or ending dates for the product team. Furthermore, budget and on-time milestones were not controlled by the team in our investigation. A redesign of processes and end-to-end service responsibility is important for cross-functional DevOps teams (Balaji et al., 2011).

3.5 Conclusion for DevOps Implementation

Our findings present insights into the DevOps implementation. We depict different characteristics and types of DevOps patterns across our case-study participants. For the implementation of the DevOps concept, it is important to convince every stakeholder. We present an overview of two key patterns that we identified with the help of our case study. The first pattern consists of operations team members with a highly internal platform orientation with the aim to integrate self-services for development teams. The second pattern presents evidence that DevOps teams have high general knowledge among team members with developer backgrounds. The teams are responsible for one or more product-oriented services.

The stakeholders should be integrated into strategic decision-making processes. The integration of DevOps teams is accompanied by several challenges. The DevOps teams need more freedom for decision-making processes and higher self-responsibility for their service. Thus, traditional hierarchies should be broken down. Organizations should only give a minimum number of rules and regulation, e.g., for technology choices. Traditional silo-organized IT departments with highly specialized knowledge need to be reorganized. Cross-functional (service-centric) teams with general knowledge about the service should be integrated within the IT department.

3.6 Acknowledgment

This research has received funding from the German Federal Ministry of Education and Research (BMBF) under grant code 03FH005PX4. The responsibility for the content lies with the authors.

4. Are You Ready for DevOps? Required Skill Set for DevOps Teams

Title	**Are You Ready for DevOps? Required Skill Set for DevOps Teams**
Authors	Wiedemann, Anna[1] (anna.wiedemann@hs-neu-ulm.de) Wiesche, Manuel[2] (wiesche@tum.de) [1]Neu-Ulm University of Applied Sciences Center for Research on Service Sciences Wileystraße 1 89231 Neu-Ulm, Germany [2]Technische Universität München Chair for Information Systems Boltzmannstraße 3 85748 Garching, Germany
Outlet	European Conference on Information Systems, Portsmouth, UK, 2018
Status	Accepted
Individual Contribution	Literature research, data collection, data analysis, identification of results, and manuscript writing

Table 22. Fact Sheet Publication P4 (Own Depiction)

Abstract

In order to react quickly to changing environments and build a customer-centric setup, more and more organizations are deciding to work with the agile IT software development approach. But for the fast delivery of new software features in very short time, other parts of the IT department are necessary as well. Hence, the DevOps concept appears and connects development to IT operations activities within service-centric IT teams. To date, there has been very little empirical research on the skills required for the successful setup of a DevOps-oriented IT team. This study addresses this gap by conducting a multi-perspective research. We have collected data with the help of a workshop and interviews with IT experts. Seven skill categories—full-stack development, analysis, functional, decision-making, social, testing, and advisory skills—with 36 concrete skills were identified. Our study highlights that a combination of distinct development, operations, and management skills is necessary to successfully work within a DevOps team. This research explains core DevOps skill categories and provides a deeper understanding of the skill set of an ideal DevOps team setting. We describe these skills and skill categories and list their implications for research and practice.

4.1 Introduction

Digital transformation has forced organizations to rethink their requirements for IT functions. The appearance of new digital technologies presents incumbent firms with changing opportunities on the one hand and major challenges on the other (Sebastian et al., 2017). These organizations have to think about how they can compete in the digital economy. Many old companies, which are at an early stage of digital transformation, are considering the capabilities necessary to develop themselves into digital leaders (Ross et al., 2016; Sebastian et al., 2017). IT organizations are under pressure to work with new technologies, improve their business understanding, enhance speed for delivering new products, and organize themselves to be more customer-centric (Bharadwaj et al., 2013; Jöhnk et al., 2017). Consequently, more and more organizations are deciding to implement agile IT setups to foster innovative capabilities and agility (Lee et al., 2015).

To address the current challenges and enable future-proof IT setups, it is essential for IT organizations to implement agile IT teams (Jöhnk et al., 2017). Further, digital transformation fosters the development of new capabilities within the IT function as well. Agile software development is limited to the collaboration between business and software development (Lwakatare et al., 2016; Sebastian et al., 2017; Tripp et al., 2016). For the fast delivery of new software features and to enhance business value, agility has to be scaled to other parts of the IT function that are necessary for the service delivery lifecycle as well (Fitzgerald & Stol, 2017; Matook et al., 2016; Ross et al., 2016). Hence, companies are transforming their IT units and implementing service-centric IT teams—the so-called DevOps teams (Sebastian et al., 2017). DevOps is an acronym of development and operation (Lwakatare et al., 2016). DevOps combines the activities of software development and operations in one team and enables the continuous deployment of new features in short cycle times (Sebastian et al., 2017). Hence, response times for customer needs are drastically reduced through the better coordination of work (Lwakatare et al., 2016).

DevOps is a rising trend and is expected to become a competitive necessity. More and more IT organizations are deciding to implement the DevOps concept (Ross et al., 2016). According to a Gartner survey conducted in 2016, about 25% of 2,000 global IT companies will work with the DevOps concept and tools according to the DevOps toolchain (Rivera & van der Meulen, 2015). Hence, DevOps is recognized as an important issue in the area of software development. Innovators in the digital area, like Amazon, have already adopted the DevOps approach and are now able to provide new codes to customers in seconds. It is clear that organizations have to develop DevOps capabilities to stay competitive through the delivery of rapid innovations (Sebastian et al., 2017). But the implementation of the DevOps concept entails extensive efforts by IT leaders to build and create new values by linking existing capabilities with novel ones (Ross et al., 2016). However, for many organizations, it is not clear which skills are needed to fulfil the different tasks that are expected from a DevOps team. Lack of skills in a project environments is one of the key factors for failure of IT projects (Keil, Lee, & Deng, 2013). A distinct skill set is highly correlated with a successful project outcome (Napier et al., 2009); hence, it is necessary to understand the required skills of DevOps teams. Despite initial intentions to investigate forms of the DevOps concept (Lwakatare et al., 2016), there is very little empirical research on the skill set of successful DevOps teams. The goal of this study is to present an empirical skill set framework for DevOps teams. Hence, with this research, we wanted to answer the following research question: *What is the ideal skill set for successful DevOps teams?*

We conducted a case study with 10 participants as a pre-study and recognized that there is the need for a common framework about the key skill set that should be present within a DevOps team. This motivated us to set up a workshop with nine experienced IT consultants in the DevOps field, moderated by two researchers, to achieve deeper insights of the DevOps phenomenon. Subsequently, we conducted further interviews with IT consultants to verify our results. To this end, we have identified skill categories that describe the ideal DevOps team's skill set. The following sections of this paper explain the DevOps phenomenon and the concept of skills. Additionally, we focus on the presentation of the workshop findings and give a description of skill categories and their characteristics. Finally, we discuss our findings and conclude with implications for theory and practice.

4.2 Theoretical Background

4.2.1 DevOps teams

To handle the complexity of rapidly changing environments, IT organizations are deciding to implement product-oriented agile IT teams. The ability to adapt to new situations is a key requirement of high-performing software development teams (Burke, Stagl, Salas, Pierce, & Kendall, 2006). In changing environments, teams must be able to react fast and with precision. During the last few years, the use of agile software development methods has gained high popularity in a rising number of organizations (Tripp et al., 2016; West et al., 2010), primarily due to higher customer satisfaction ensured by the provided software as well as better software quality and higher project success (Maruping et al., 2009; Tripp et al., 2016). But research and practice highlight that a tighter collaboration between the development and the operations part of an IT function is necessary to ensure that errors are quickly fixed and the quality and resilience of the software are enhanced. DevOps can help to combine these approaches and reduce software cycles by pushing new code quickly into the production environment (Fitzgerald and Stol 2017). For the rapid delivery of new software features, innovations, quick handling of problems, and integrating maintenance activities, IT departments should use cross-functional teams rather than separated silo architectures (Fitzgerald & Stol, 2014).

Prior research shows that agile software development is limited to the area of development activities. Therefore, in a lot of organizations, the agile approach does not go beyond development and customer-gain deployments of new software features except in rare cases (Lwakatare et al., 2016). One reason for this is the poor communication and collaboration between developers and operations personnel. To address this issue, the DevOps approach enables the scaling of agility to the entire company (Gotel & Leip, 2007). The aim of DevOps is to foster collaboration, automation, virtualization, and tools to bridge the activities of software development and operation (Humble & Molesky, 2011; Lwakatare et al., 2016). Through DevOps, solutions are delivered to avoid interruptions between different stages of the complete software delivery process (Fitzgerald & Stol, 2014). The entire software development life-cycle comprises the steps of planning, development, and operation tasks. DevOps helps companies to acquire the necessary speed and flexibility to ensure the constant and rapid development and implementation of digital innovation (Ross et al., 2016). Hence, risks associated with software releases can be reduced and feedback for new software features is received faster. In addition, agile methods can be used to manage the software development part and meetings of the team (Lwakatare et al., 2016).

From a practical point of view, the state of DevOps report shows that high-performing DevOps organizations are able to deploy 200 times more frequently than low-performers. The change failure rate is three times lower than in traditional settings. Further, DevOps teams spend 22% less time on unplanned work and rework and thus are able to spend more time on new feature development (Brown et al., 2016). Software development comprises the analysis of new requirement, design and coding activities, and verification and testing, whereas software operation focuses on maintenance and software installation tasks (Fitzgerald & Stol, 2014). Within a DevOps team, a very broad understanding of skills is necessary. Whereas traditional silo-oriented IT functions foster specialized knowledge in one area, e.g. departments, DevOps teams need a very broad generalist setup over all tasks of the software delivery lifecycle for which they are responsible (Rouse, 2016). Hence, in cross-functional IT teams, the same activities need to be implemented for service delivery as in traditional IT functions, with separated units of specialists.

4.2.2 Concept of skills

Skills or competency models are used to describe the abilities, knowledge, attitudes, and traits that are essential for the effective performance of an organizational position or role (Mansfield, 1996; Matook & Maruping, 2014). Usually, the term competency is used synonymously to skills, but there are fine differences. Skills denote the ability to finish tasks and solve problems through the application of knowledge, whereas competencies represent the ability to use skills, knowledge, and other personal or social capabilities to assess situations or to work in professional and personal environments (Boehm et al., 2013; Peppard, 2010). Skills outline the abilities and behavior needed to competently perform activities and tasks that are expected by the DevOps team (Matook & Maruping, 2014). Skill models are helpful for measuring the performance of people to find out if they need personal development or need to improve effectiveness. Furthermore, they support the recruitment and staff planning process (Lucia & Lepsinger, 1999; Matook & Maruping, 2014). Competencies are used in different research areas (Klendauer et al., 2012). A prominent example is the PMI's Project Competence Development framework, which states that project management competencies have three components—a personal, a performance, and a knowledge component (Institute, 2002; Napier et al., 2009). Furthermore, prior literature describes skill requirements for IT project managers and fosters the understanding of the several skills that lead to successful project management. Napier et al. (2009) identify nine skill categories for IT project managers, including two skill categories that have rarely been discussed before. With regard to agile software development, Matook and Maruping (2014) present a competency model for customer representatives that was derived with the help of interviews. Ten competencies are presented and grouped within three core areas. However, we found little empirical evidence on the matter of skills needed by DevOps to perform successfully.

Skill models are used for different reasons, such as for the development of a job model or adapting existing skill models for similar job profiles. When starting from the scratch, deep knowledge, e.g. from experts, is necessary for generating suitable results (Matook & Maruping, 2014). In DevOps teams, little explanation of job roles exists; it is a concept for collaboration. Hence, the aim of this research is to focus on the general DevOps team's skill setup. A diverse skill set helps team members to develop new perspectives, ideas, and creative outcome (Lee & Xia, 2010; Matook et al., 2016). There are two types of competencies—performance-based and attribute-based. Performance-based competencies focus on skillful practice. This type of competency depicts the job know-how of the individual (Crawford, 2005; Napier et al., 2009). Competent DevOps team members can be identified with the help of effective actions that lead to successful outcome. Attribute-based competencies focus on skill-as-attributes, i.e. on the degree to which the individual has the necessary set of knowledge that a DevOps team should acquire. Attribute-based competencies are characterized by the know-what (Klendauer et al., 2012; Napier et al., 2009) of DevOps team members. Through skills-as-attributes perspectives, insights into requirements of DevOps teams for successful practice can be obtained.

The skillful-practice method addresses the behaviors, actions, and activities expected by the employers (Crawford, 2005). This is helpful for the investigation of activities and interactions of effective project management. On the other hand, the skills-as-attribute method concentrates on explicit and necessary knowledge, skills, and personal characteristics (Napier et al., 2009) of DevOps team members. This method is helpful for investigating required skills based on contextual aspects, measuring the relationship between the skill set and outcome of projects. For practical purposes, it is helpful to understand which skills should be considered when a DevOps team is set up. These skills can be identified through different research methods, such as with the help of qualitative methods like expert interviews, as done in prior research (Klendauer et al., 2012; Napier et al., 2009).

From the perspective of agile development, literature does not deliver a broader mix of skills (Matook & Maruping, 2014); hence, there is a need for skill models in the area of DevOps-oriented working. For the purpose of this research, we have chosen the skill-as-attribute perspective for investigating the DevOps team members' skills. We want to provide deep insights into the skill set of DevOps teams and consider skills as mandatory for the successful performance of a DevOps team.

4.3 Research Method

To answer our research question, we conducted a multi-perspective research approach (Boehm et al., 2013). Therefore, we decided to undertake a comprehensive multiple-case study among companies that are already working with the DevOps approach. The qualitative instrument of the multiple-case study with 10 cases can be seen as the pre-study, which delivers the basis for a workshop with IT experts of one IT consultancy company. Workshops are a good method to study new and arising phenomena, because they allow discussion about the answers of the participants in several rounds (Boehm et al., 2013). Afterwards, three interviews with IT consultants were conducted for verification. Hence, in this paper, we use the workshop method and subsequent interviews to identify and verify the key skill set of DevOps teams.

For the case study, we contacted different firms from different industries via email and telephone. Ultimately, 10 organization agreed to participate with one DevOps team in the case study. The case study approach is defined as "an empirical inquiry that investigates a contemporary phenomenon in depth and within its real-life context" (Yin, 2009, p. 18). Since the phenomenon of DevOps is quite new and unexplored (Yin, 2009), the case study approach is appropriate. The advantage of case studies is that they can zoom in on real-life situations and test or develop theoretical perspectives in relation to certain phenomena as they unfold in practice (Flyvbjerg, 2006). For doing so, we were able to investigate a row of skills and skill categories implemented within a DevOps team. An interview guideline not only helped to keep the interaction focused as data collection proceeded, it was also used to ensure comparability of the interview data across individuals, settings, and researchers (Maxwell 2013). Each interview lasted about 45–75 minutes and was mainly conducted through face-to-face meetings or by telephone. At least one manager (e.g. team lead) and a person concerned with daily business tasks were interviewed. Selecting both managers and team members enabled us to gain knowledge from a leadership and an operational perspective. Every interview was recorded, transcribed, and analyzed. Moreover, a comprehensive number of notes was taken during the interviews. With the help of the pre-study, we found that there is a lack of common understanding about the skills that are necessary for a DevOps team. Hence, we decided to conduct further investigation to obtain a more detailed picture about DevOps skill set for research and practice.

With the following exploratory study, we built up on the multiple-case study approach and investigated DevOps team members' construction within an ideal setting. The pre-study was the trigger for deciding to organize a workshop with IT consultant experts. We recognized that there are a lot of different skill sets in the investigated DevOps teams but a lack of a common understanding of a necessary skill set. Hence, we decided to build up on these findings and organize a workshop with IT experts. Therefore, we contacted an IT consultancy firm that is familiar with consulting projects pertaining to DevOps implementation. After a preliminary talk with two consultants, we reached an agreement about the content and identified participants for the workshop. Subsequently, we were able to conduct a personal DevOps expert workshop with nine IT consultants in October 2017. The IT consultants have different backgrounds, but are all working with DevOps topics. Our aim was to get a 360-degree view on the topic with the help of the workshop (Boehm et al., 2013). Figure 5 presents the workshop settings with the nine IT consultants. The workshop was moderated by two researchers. In sum, the figure presents the gender, working experience (years), current job role, and consultancy area of the IT consultants, namely manufacturing or digital business solutions (DBS).

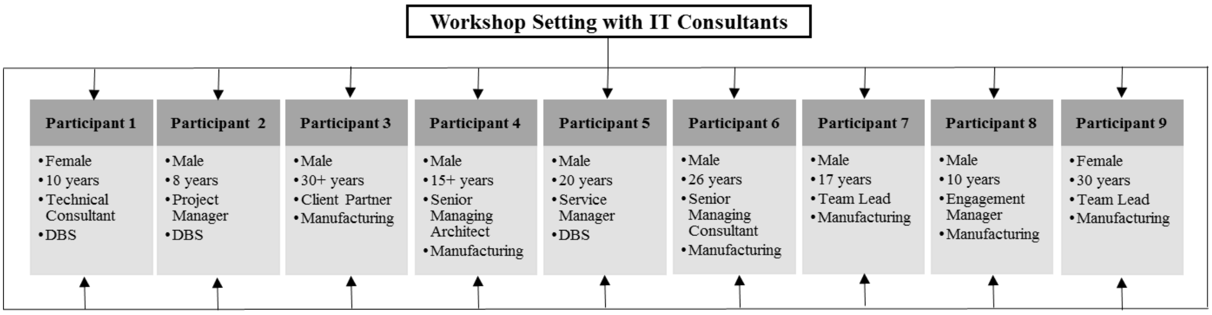


Figure 5. Workshop Participants' Demographics (Own Depiction)

As far as possible, workshops should start from a blank sheet of paper; issues and ideas should emerge through collaboration between practitioners and researchers (Howard et al., 2004). Thus, the findings of the workshop can be cross-referenced against the findings of preliminary case studies and the follow-up interviews and vice versa. The workshop meeting had a duration around two and a half hours. The researchers started with a short introduction and explained the goal of the workshop. To avoid bias, the researchers focused on the process of data collection, so as to moderate the workshop without influencing the opinions of the participants (Howard et al., 2004). In the workshop, the participants were asked to elaborate the skills required by a DevOps team in an ideal setting. We asked the experts to write down on cards the key skills that they see within an ideal DevOps team. Afterwards, the experts were asked to present their findings and pinned the cards with the key skills on a wall. Every expert expanded the wall with skills that were not mentioned before and discussed the findings with the others. Skills that were mentioned by more than one participant were prioritized. The skills were discussed in the group until no new insights into the topic could be gained. Hence, we had an iterative approach where skills were discussed and the findings were extended and verified by every presentation of skills by the IT consultants. The presentation of the findings and the discussion were recorded, transcribed, and photographed. The findings of the workshop were sent to the participants and we requested them to respond with feedback and verification within one week.

After conducting the workshop, we interviewed three other IT consultants of the company (Table 23) who were not participants of the workshop and asked them for the ideal setting of skills that should be implemented within a DevOps team. The aim was to verify and extend our findings. The interview partners could not be allowed to be influenced by our prior findings. Hence, we showed them the results of the workshop in the second half of the interviews. In the first half of the interview, the participants were asked to explain the skills they see in an ideal DevOps team. Then we asked the interviewees to supplement and comment on the results of the workshop. Thereafter, we presented an overview of the achieved skill set of the workshop. These interviews provide some extensions and verifications of the workshops results. After three interviews, no new insights were given by the IT experts. All interview data and the workshop data were coded using the software NVivo 10 (Kude et al., 2014).

	Gender	Working experience	Job title	Industry
I1	Male	21 years	Managing Technical Consultant	Digital Business Solutions
I2	Male	29 years	Managing Senior Consultant	Financial Services
I3	Male	17 years	Team Lead	Manufacturing

Table 23. Interview Participants' Demographics (Own Depiction)

With the help of the workshop, an iterative approach for generating a skill set was used. As mentioned before, necessary skills were written down on cards and pinned to the wall. All nine participants contributed detailed information about why they felt a skill is important in a DevOps team. Furthermore, we went through transcripts of the follow-up interviews and coded them as well. We started the coding process following the guidelines given by Miles and Huberman (1994). Hence, we began with an open coding process. During the coding, the research team took notes to justify the coding section. Afterwards, the results were analyzed regarding key skills of DevOps members. Then, the research team compared its findings for each dimension to identify commonalities, relationships, and patterns. The focus was placed on the identification of the categories from the pre-study and the amplification and extension of the workshop results. Furthermore, we focused on the concrete skills that emerged during the data analysis of the expert workshop. Figure 6 depicts our research approach.

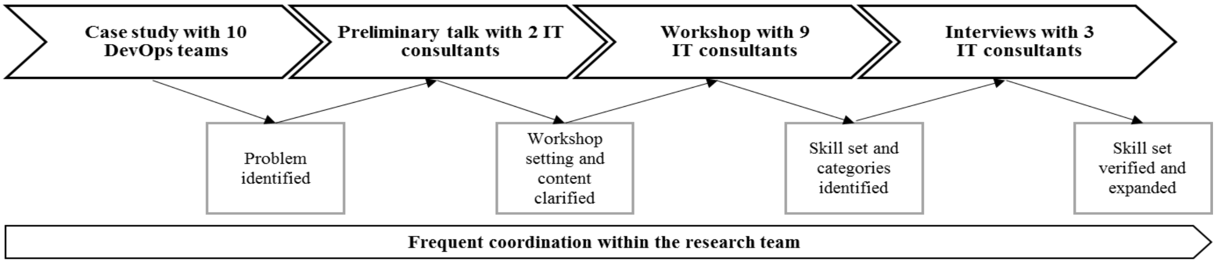

Case study with 10 DevOps teams
Preliminary talk with 2 IT consultants
Workshop with 9 IT consultants
Interviews with 3 IT consultants
Problem identified
Workshop setting and content clarified
Skill set and categories identified
Skill set verified and expanded
Frequent coordination within the research team

Figure 6. Research Approach (Own Depiction)

During data analysis, the research team was able to derive key skill categories as topics for further subdivision into concrete skills to build a skill set. The concrete skills for ideal settings in DevOps teams were identified through the expert workshop and three post-processing interviews.

4.4 Findings

Our findings present a consolidation of the raw data with the same underlying ideas regarding skill categories, similar to the studies by Napier et al. (2009) and Klendauer et al. (2012). During the workshop, the participants mentioned some main categories or elements, and it was noted if similar detailed skills were mentioned by them, e.g., development skills. Additionally, the post-workshop interviews not only verified the skills but also automatically built core categories to sort the skills into them. After transcribing the audio data, the workshop and post-workshop interviews were coded regarding skills and core categories. Further, we extensively discussed the skills and categories in the research team, Finally, we were able to present seven core skill categories with 36 concrete skill items. The findings of our interviews for the pre-case study give insights into capabilities and skill categories of DevOps. The informants of the pre-study were unable to explain detailed skills and there was a lack of common understanding about which key skills seem to be necessary within a DevOps team. Hence, we decided to close this gap and present a common skill set for an ideal DevOps team to foster the general understanding of DevOps.

The several skill categories are now explained. The first one is **full-stack development**. We identified seven concrete skills within this category. During the pre-study, several cases highlighted that all activities of the software delivery lifecycle should be conducted by the DevOps team. The workshop participants highlighted that the comprehensive understanding of the different layers, tiers, and platforms is essential to work in a DevOps team.

Next, **analysis skills** are very important for DevOps team members. Analysis skills encompass IT operations skills (Gordon & Gordon, 2002) that should now be learned by the complete cross-functional team. This is essential for monitoring, problem analysis, and management tasks. Further, all team members need skills for problem abstraction to determine where the problem could come from.

The next category we identified is **functional skills**. This category includes essential functional and consulting skills of the DevOps team. Functional skills are necessary to understand the business needs and achieve the desired benefits and goals (Liu et al., 2015). The team is now responsible for a service and thus has to manage it. This encompasses budgeting skills, work and time planning, and a deep understanding of the complete service process.

Decision-making skills are identified as another core category. DevOps members must be able to decide quickly, be self-confident, and take over responsibilities for possible failures. Decision-making is characterized by the identification of an action, the decision to take an action, determining when the action is finished, taking over the responsibility, and supporting this action (McAvoy & Butler, 2009). In traditional IT organization, decision-making is a longitudinal process (Horlach, Drews, Schirmer, & Boehmann, 2017; McAvoy, Nagle, & Sammon, 2013) and now the team should be able to take over these skills for a concrete service.

The fifth category is **social skills**. These skills comprise the ability and willingness to share knowledge and communicate within the team. The IT consultants highlighted that a constructive feedback culture is essential for building a working relationship in DevOps teams (Wagner et al., 2014). The willingness of the members to actively learn from one another enhances their knowledge of technology, users, and team activities. Hence, better software quality and performance can be achieved (Spohrer et al., 2012).

Testing skills are necessary to understand the general software-testing tasks needed to provide a new software feature. For avoiding software failures, software testing is an essential activity. With the help of software testing, the correctness of the code can be proved, security issues identified, and the quality of the software guaranteed (Onita & Dhaliwal, 2011). For DevOps team members, it is necessary to implement testing skills, since new software features are delivered in very short time with the help of a high degree of automation (Fitzgerald & Stol, 2017).

Advisory skills were identified as the seventh core category, because the workshop participants highlighted that there must be an awareness of the commercial consequences of breaking service level agreements (SLAs) or other agreements and commitments. Thus, team members have to know these agreements. DevOps team members have different backgrounds and are not always used to working with SLAs, especially in consultant projects. If there is a failure, people must work to solve the problem as fast as possible (Trusson et al., 2014). A breach of any agreement should generally be avoided. The following Table 24 presents an overview of the identified skill categories, their description, and the identified key skills.

Skill Category	Description	Skills
Full-stack development skills	Full-stack development skills are very broad and encompasses the complete stack and interests in all software technology. This is because the DevOps team is responsible for the service delivery lifecycle. Hence, familiarity with all layers of the software and platforms is present in the DevOps team.	• Learn overarching development abilities • Know the architecture of the software system • Acquire extensive understanding of the platform • Build platforms independently • Understand non-functional requirements • Know and understand tools and supporting processes • Know and understand continuous integration/development/delivery
Analysis skills	Analysis skills mostly concern IT operation tasks like monitoring the system and the team's ability to comprehensively understand problem management (Gordon & Gordon, 2002).	• Identify and resolve problems • Monitor the system operations • Analyze code and network problems • Abstract problems
Functional skills	Functional skills help to understand processes from a technical as well as from a business consultant view. This encompasses, apart from the technical knowledge about the service, a structured mode of operation and consulting the customers with regard to time and budget.	• Understand the complete processes • Understand the application • Build technological knowledge • Perform tasks in a structured way • Create consulting concepts • Record and understand customer demands • Build commercial project understanding
Decision-making skills	Decision-making skills address the fast and individual decision-making process (McAvoy et al., 2013). The team has to combine them with a degree of self-organization and willingness to take responsibility.	• Take responsibility for actions and wrongdoing • Encourage appropriate decision-making • Make decisions quickly • Enable self-organization • Develop dissolving power

Social skills	Social skills pertain to the interaction within the team as well as with the customer. Thereby, the team is able to collaborate and develop working relationship for successful working (Wagner et al., 2014).	• Approve feedback ability • Communicate according to target group • Learn intercultural communication skills • Show willingness to learn new skills • Work reactively and proactively within the team • Show willingness to share knowledge • Show willingness to learn from one another
Testing skills	Testing skills ensures that the software and new features work. Testing is necessary to ensure that a software meets the specific requirements (Onita & Dhaliwal, 2011).	• Understand test automation functions • Automate tests • Understand functionalities for test management
Advisory skills	Agreements within a service function help to improve collaboration. These agreements need to be understood and met (Trusson et al., 2014).	• Understand agreements • Understand the commercial impact of agreements • Comply with agreements

Table 24. Skill Categories, Definitions, and Skills for DevOps Teams (Own Depiction)

4.5 Discussion

This paper is one of the first to elaborate a skill model for DevOps members; hence, it contributes with insights in the area of software development and operations. In this section, we discuss the identified DevOps-related skill categories. The order of the presented skills and categories reflects their importance, as mentioned by the workshop participants and interviewees. For example, full-stack development skills were mentioned by every participant of the workshop.

Full-stack development skills: This category was mentioned as the most important by the workshop participants. These skills visualize the development perspective of a DevOps team. For the setup of a DevOps team, a comprehensive knowledge of development activity is necessary. DevOps team members need to broaden their development skills to have comprehensive knowledge about the application. Hence, skills pertaining to the different layers, tiers, and platforms are necessary. Thus, full-stack development skills are necessary *“so that it makes sense if the application at any point faces problems [...] I may have some issues with the application, but it's actually because my cloud infrastructure is not working properly” (8). I*t is not sufficient for a DevOps team member to have only frontend or backend development knowledge. If a person is responsible for solving problems in a very short time, comprehensive technical knowledge is necessary. In traditional IT functions, silos for development with members who have very highly specialized knowledge in one area are used (Horlach et al., 2017).

Analysis skills: The analysis skill category encompasses IT operation activities in particular and hence, comprises the Ops of DevOps. Analysis skills with focus on problem-solving are necessary, since the DevOps team is responsible for the complete service, and if a problem appears, it has to be solved. Problem-solving skills are critical within the software delivery process and developers have to adopt this knowledge to enhance performance (Ozer & Vogel, 2015). Key skills for incident and problem-solving creativity are necessary (Trusson et al., 2014). The members have to wear pagers and should be willing to do night-support activities. Within an ideal DevOps team, every member is able to contribute with analysis skills. Analysis skills to manage problems and the corresponding skills are essential for the project success. IS development projects are connected with knowledge-intensive tasks including technical and application-based domain knowledge (Lin, Chen, Hsu, & Fu, 2015). Hence, there is a need to understand the analyzing skills within the DevOps team. If team members have to do nighttime support, all members need to have the ability *"to identify from where the problem comes and analyze or solve problems so that you can at least fix them till the next day" (8),* even if it is a person with no strong IT operational background.

Functional skills: The functional skill category includes comprehensive know-how about the complete service delivery process as well as consulting skills for collaboration with the customer (Blohm, Bretschneider, Leimeister, & Krcmar, 2010). *"Process know-how about the development, operation, and business process is necessary" (8).* An ideal DevOps team must be clear about the functional processes. DevOps team members *"need functional knowledge about the application, or the team has to know the application from a functional perspective and should be able to assess it technically" (8).* A comprehensive knowledge of business processes and the ability to plan workloads and budget are essential in DevOps teams. Literature highlights that deeper understanding of the business functions is important for successful project collaboration. By including management and business knowledge into the team, solutions are delivered to avoid interruptions between different stages of planning, building, and running these stages (Fitzgerald & Stol, 2017). Ideal DevOps teams are able to consult the customer, plan their working efforts effectively, and have a deep understanding of the development, operations, and business processes.

Decision-making skills: Decision-making skills refer to the ability of DevOps team members to make fast decisions, take responsibility if there is any failure and organize themselves within the team. The importance of decision-making skills are mentioned in existing literature with regard to agile software delivery (McAvoy & Butler, 2009). Within an ideal DevOps skill setting, the team members are able to take over *"self-responsibility, enjoy decision-making, and adopt responsibility" (4).* Especially in the IT operations area, a lot of ad hoc decisions have to be made in *"stressful situations, when a major incident appears and a system is down" (5).* Usually, developers are not confronted with this fast decision-making process. Thus, organizations are under pressure to enhance the mindset and skills of developers so that they can fit in a DevOps environment. The members must *"keep calm and work with dissolving power" (I1)* when pressure from external environments appear or a major incident occurs, for example. Thus, within ideal DevOps teams, members must be able to react quickly to uncertainties and create an *"optimally organized" (I2)* team with a structured working mode.

Social skills: Social skills are necessary for interactions within the team as well as with the customer. Hence, connections between network actors, common language and code, and shared narratives, as well as the assets that create and leverage relationships should be investigated

(Liu et al., 2015; Nahapiet & Ghoshal, 1998). Social skills include skills like trust and knowledge-sharing, which are essential in ideal DevOps teams. The team members have to achieve a *"feedback ability, in both directions—so, to give feedback but also to accept it" (6).* This is connected with strong communications skills. Team member must be able to communicate in an audience-oriented manner. Usually, in traditional IT departments development and IT operations, people do not have a strong communication process (Horlach et al., 2017). In an ideal situation, the team members communicate, trust each other, and build a culture of communication. Furthermore, an end-to-end flow with customer demand is implemented (Fitzgerald & Stol, 2017) and the DevOps team members must consider this in their communications skills as well.

Testing skills: The DevOps movement is related to a high degree of automation (Fitzgerald & Stol, 2017), e.g. test automation. Continuous testing includes the automation of some test processes and defines the priority of test cases with the aim to reduce the time between implementation and detection of errors and removal of root causes (Fitzgerald & Stol, 2014). Thus, ideal members of DevOps team have an understanding of software testing, which includes *"special field testing" (I3)* and know-how to automate tests and manage test automation. Software testing includes late-stage system testing on a complete and integrated system to assess compliance with the requirements (Onita & Dhaliwal, 2011). Team members need testing skills because *"it is a very important topic that should be addressed early so that people can manage the tests and the automation" (5).* In summary, comprehensive testing skills are important in ideal DevOps teams.

Advisory skills: The participants of the workshop and the interviews highlight that an understanding of SLAs and their commercial impact is a key skill for DevOps teams. Not every person in the IT is familiar with SLAs or other agreements. Hence, DevOps team members have to understand the impacts agreements, because the vendor has to pay fees in case of non-compliances (Hoermann, Hlavka, Schermann, & Krcmar, 2015): *"that you understand how SLAs work. But the developers often do not have that understanding" (6)*. Team members with development background are not used to working with SLAs and need an awareness for them. Ideal skilled team members of DevOps environment can effectively react to ensure that the SLA is met and no monetary damage is incurred.

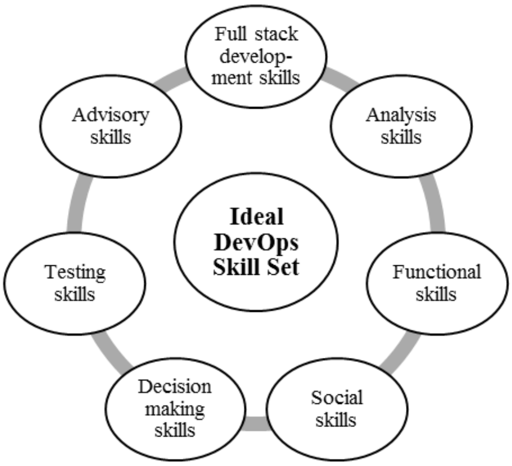


Figure 7. Skill Categories of Ideal DevOps Teams (Own Depiction)

In summary, we present a skill set for ideal DevOps team members with seven core categories and 36 skills. Several existing research works focus on requirements management skills (Chakraborty, Sarker, & Sarker, 2010; Klendauer et al., 2012), customer representatives in agile projects (Matook & Maruping, 2014), and IT project management skills (Napier et al., 2009). With the help of this paper, we contribute to existing literature by investigating the new phenomenon of DevOps. Compared to competencies in agile software development as depicted in existing research, e.g. Matook and Maruping (2014), this study shows DevOps skill sets by depicting the necessary operations skills like analysis skills and functional skills. Social-rational skills and development skills are also important in agile team settings, but the maintenance processes are usually neglected. We present a skill set for cross-functional DevOps teams that shows that skills from several areas—development, operations, and business—are important to work as a DevOps team.

IT organizations are confronted with the challenges of implementing the same skills in a very small cross-functional IT team. Therefore, a team of generalists is necessary. Team members have to leverage their knowledge in several areas in order to develop quality software (Spohrer et al., 2012). Workshop Participant 4 mentioned that *"at the current job market, I can't see where we can find such highly skilled people."* Furthermore, organizations need to build incentives that go beyond money to find well-educated people *"if you are not Facebook or Google" (1).* That means that not every company is able to pay above average salaries to recruit this highly skilled people. Hence, organizations have to rethink their development programs for employees in order to train their existing IT staff with these full-stack development skills.

During the analysis of our findings, we found that these categories can have overlaps and dependencies that should be considered. For example, decision-making skills and social skills have dependencies, because the willingness to learn new skills helps the team members to develop a strong dissolving power. Furthermore, software development and testing are major parts of the software delivery lifecycle and have to be aligned to foster business success (Onita & Dhaliwal, 2011). With the help of an ideal DevOps setting, these activities are embedded in one team. Every team member should have these skills to a certain degree. Furthermore, we have defined our categories with regard to the importance of the skills, as mentioned by the workshop participants and interviewees with the help of discussions within the research team.

Furthermore, the ideal DevOps setting is dependent on several issues, one of which is governance. If IT organizations work with third-party providers or contracts, a deep understanding of SLA is necessary. If your IT functions as an internal service provider for a business section, the focus will not lie on a strong SLA process but rather on other internal agreements. Within an internal DevOps organization (without providers), *"SLAs are quite complex"* because of the high *"process character"* that should be avoided in general within an agile DevOps environment (I1). Strong process orientation is typical for traditional IT functions. Within agile-oriented IT functions, the more processes are defined, the more agility is restricted (Horlach et al., 2017). DevOps approach enables the scaling of agility to the entire company (Gotel & Leip, 2007). We have included advisory skills within our research since we talked to an IT consultancy firm; SLAs and contracts were reported to be foundations of their daily work and were mentioned several times in the workshop as well as in the individual expert interviews. Hence, priorities of the skill set will be different, depending on the conditions of organization and the structure of service delivery. Furthermore, implementing concrete DevOps skills within a team is strongly dependent on the IT function and the overall organizational management.

The full-stack development, analysis, and functional skills are rarely discussed in existing literature. These skills were assessed with the highest priority. The combination of roles that are responsible for planning, building, and running within a team has been emphasized for a long time (Markus & Keil, 1994). Within an ideal DevOps team, every member should be able to undertake large parts of all these activities. Since this is a new approach and a great challenge for firms, our research provides a skill set that is helpful to identify and integrate skills to start implementing the DevOps concept.

However, organizations are confronted with the challenges of digital transformation and have to adapt agile-oriented working modes like the DevOps concept. This presents a tradeoff in reality, since the necessary skill set introduces immense requirements regarding IT staff that cannot be easily found on the job market or existing silo-oriented IT function. There is only a *"short training period into a high level of complexity" (I3)* of tasks. Hence, organizations have to take great efforts to train the personnel with the presented skills to develop competitive capabilities in the area of digital transformation.

4.5.1 Implications for theory and practice

In summary, our research represents a major contribution to research and practice. We deliver a framework with empirically grounded insights into the skills needed by an ideal DevOps team. These identified skill categories are crucial within this ideal DevOps team setting and essential for a successful software delivery lifecycle. Hence, we deliver an initial understanding of the attribute-based competency perspectives of DevOps team members and provide a new research in this area. The present research sheds light on a DevOps team profile of an IT function that has not been systematically analyzed in depth before.

In summary, our research highlights the specific profile of DevOps teams and presents a framework that is helpful for the individual team members as well as for the organization to develop core skills. Closer collaboration is necessary between the core activities of DevOps, development, and operations. We identify DevOps-related skill categories of full-stack development, analysis, and functional skills as the most important. This shows that IT personnel with development as well as management and IT operation backgrounds have to expand their knowledge about these areas. Thus, with the help of the presented skills, we deliver concrete insights into team settings where planning, building, and running functions are integrated. We enhance the existing literature, which emphasizes the building of teams including business, development and implementations perspectives to foster performance (Markus & Keil, 1994). Furthermore, the willingness to take self-responsibility and make decisions by the self-organized DevOps team and not by the management is a fundamental skill within an ideal DevOps team setting. This research focuses on the skills-as-attributes; how these skills affect the performance of DevOps teams is not in the scope of this study and should be considered by further research.

This research is the first step toward enhancing the understanding of DevOps skills, as mentioned in prior literature (Sebastian et al., 2017). We sort the identified skills into seven categories and hence provide a basis for further research. Further, we present differences and dependencies in existing research with regard to skills. Future studies can examine how these skill categories influence DevOps outcome. Moreover, interrelations between the skill categories and the several skills can be investigated. This study presents a starting point for an in-depth investigation of the field to enhance the theory.

The research presents interesting findings for practice as well. We deliver insights into how an ideal DevOps team can be set up. A set of key skills is provided by this research that can help IT managers to develop and integrate a DevOps teams. Our research provides a comprehensive overview of the skill that should be considered in DevOps team settings. We provide guidelines for the implementation of DevOps skills. Additionally, we give recommendations for designing trainings to develop highly skilled IT people. Courses for DevOps team members could be established to educate people through the organization's own IT function. This research can also be used to supervise or control project in with the DevOps approach is already used. This paper shows that practitioners increasingly need to be made aware of the DevOps concept and implement trainings and efforts to stay up-to-date. This research presents a comprehensive skill set in the case of ideal DevOps teams' setup. Since these high-skilled IT people are not easy to find or train, further research should investigate how far every team member should be able to assist in every task. Furthermore, studies should investigate how DevOps teams can be supported through the IT function. Even if the team should work autonomously, there are interdependencies with other teams or the IT function. The identification of cross-sectional collaborations with other DevOps teams or with a traditional IT department is helpful for implementing DevOps teams.

4.5.2 Limitations

While interpreting the results, some limitations should be considered. This research focuses on the perception of IT consultants of one company. These consultants have worked in different projects with various levels of work experience. A large part of this research is based on the results of this workshop, which reduces the generalizability of our research. Further research should broaden this perspective by including a complementary study with more participants to substantiate the findings or find an additional company with a similar setting. Moreover, views of other stakeholders and internal DevOps teams might be interesting and uncover other characteristics of the skill set as well and should be investigated in further research. Furthermore, we present a skill set of an ideal DevOps setting, independent of the structure, industry, management, etc. In real-life context, there might appear some differences in the priority and characteristics of the several skills. Further, this research focuses on the IT perspective because we only talked to IT people. Agile development and DevOps team settings have a high impact on customers and business; hence, the research should be expanded through a business view as well.

4.6 Conclusion

This paper contributes to the understanding of skills of an ideal DevOps team. We used the concept of competencies and skills and identify attribute-based competencies that are characterized by the know-what (Klendauer et al., 2012; Napier et al., 2009) of DevOps team members. With the help of a multi-perspective research approach that comprises a case study research, workshops, and interviews for verifications, we were able to derive an ideal skill set for DevOps teams. Our findings show the importance of skills and skill categories to build effective and successful DevOps team. Since there exists less common view about DevOps key skills, we follow the path of prior literature (Fitzgerald & Stol, 2017; Sebastian et al., 2017) to further investigate this phenomenon. This research enriches the understanding of several skills that were less defined in prior research. We identify seven core categories and report that three of them in particular—full-stack development, analysis, and functional skills—and above all the interaction between them, have not been widely discussed in prior IS literature.

Moreover, by analyzing and prioritizing our findings, this study highlights three major skill topics. This shows that a combination of strong development, operations, and management skills is necessary to successfully work within a DevOps team. Our findings expand existing literature in the field of skill as well as in the area of digital transformation—to be more concrete, DevOps and agile IT teams—and give practical implications as well. Existing research mainly focuses on agile and project management skills. The findings deliver a strong foundation for further research opportunities about the DevOps concept, possible relationships among the skill categories, several skills, and the effects of performance and outcomes of DevOps teams.

4.7 Acknowledgment

This research has received funding from the German Federal Ministry of Education and Research (BMBF) under grant code 03FH005PX4. The responsibility for the content lies with the authors.

5. Implementing the Planning Process within DevOps Teams to Achieve Continuous Innovation

Title	**Implementing the Planning Process within DevOps Teams to Achieve Continuous Innovation**
Authors	Wiedemann, Anna[1] (anna.wiedemann@hs-neu-ulm.de) Wiesche, Manuel[2] (wiesche@tum.de) Gewald, Heiko[1] (heiko.gewald@hs-neu-ulm.de) Krcmar, Helmut[2] (krcmar@tum.de) [1]Neu-Ulm University of Applied Sciences, Center for Research on Service Sciences, Wileystraße 1, 89231 Neu-Ulm, Germany [2]Technische Universität München, Chair for Information Systems, Boltzmannstraße 3, 85748 Garching, Germany
Outlet	Hawaii International Conference on System Sciences (HICSS), Maui, USA, 2019
Status	Accepted
Individual Contribution	Literature research, data collection, data analysis, identification of results, and manuscript writing

Table 25. Fact Sheet Publication P5 (Own Depiction)

Abstract

Integrating business capabilities into software development projects is still a major challenge for organizations. New ways of working are appearing in response to react to novel market places. Hence, there are more and more business managers with good IT knowledge; thus, software developers need to understand business processes. Hence, the relationship between software development, operations, and business strategy needs to be enhanced. For collecting customer perspectives in IT projects, new approaches like DevOps and BizDevOps are being used. The customer view can be integrated within software development teams through the planning processes. Our findings show that continuous innovation mechanisms are connected with the planning of customer requirements. We present planning scalability, security, and quality as rich descriptions of continuous innovation. Furthermore, we present core categories of how the customer perspectives can be integrated within a DevOps team and insights on how planning areas influence the continuous innovation mechanisms.

5.1 Introduction

Business departments are working in new ways develop and explore new markets around the world. Organizations are under pressure to integrate organizational agility and respond quickly to changing customer demands. Hence, agility is a major concern in the current business world (Holmqvist & Pessi, 2006). It can be supported through Information Technology (IT) (Lowry & Wilson, 2016). A number of trends have appeared in research and practices. For example, the concept of DevOps (Development and Operations) describes the continuous collaboration of software development and operational activities to quickly provide new software components to the customer (Fitzgerald & Stol, 2014; Wiedemann & Wiesche, 2018a).

The gap between business managers and software developers is a well-known problem (Markus & Keil, 1994) because building business capabilities is still a great challenge for IT managers (Recker, Rosemann, Indulska, & Green, 2009). IT employees like developers are often very technical and tool-oriented; they search for the easiest technical solution to problems (Wiesche & Krcmar, 2014). In the past, usually IT employees had the power to make decisions regarding the management of IT projects. But this attitude has changed because more and more business people have very good knowledge about technology. For closer cooperation, technical employees should work on their business knowledge and vice versa (Fitzgerald & Stol, 2017).

Existing research identifies the need for a closer connection between business managers and IT employees (Lehtola, Kauppinen, Vähäniitty, & Komssi, 2009). In practice, many software developers understand the need and are willing to collaborate with the business to make strategic decisions. It is suggested that stakeholders should be integrated into the software development process at a very early stage of the software delivery lifecycle (SDLC) (Fitzgerald & Stol, 2014, 2017). This integration could be achieved through planning processes. Continuous planning is an important topic in recent publications (Lehtola et al., 2009). In the past, planning was often combined in annual financial cycles in traditional software development projects with very few software releases (Chen, 2017). A failure in the traditional planning cycle might have necessitated another planning cycle to resolve the problem. While annual planning cycles are not unusual in traditional project settings, continuous planning is considered a key prerequisite in the context of agile development and for delivering fast and new software (Fitzgerald & Stol, 2017; Knight, Rabideau, Chien, Engelhardt, & Sherwood, 2001; Pflügler et al., 2018).

Existing research highlights that a lack of an efficient IT architecture may hinder enterprise agility. Monolithic IT architectures are critical for firms when adjustments to processes are necessary in response to changing demands. However, high costs may be incurred when the organization wants to integrate a new strategy, for example (Overby, Bharadwaj, & Sambamurthy, 2006). Hence, to achieve enterprise agility, DevOps could be a suitable way of breaking down software monoliths into smaller services, where the responsibilities lie with one cross-functional team [18].

The relationship between software development and business strategy needs to be continuous. Literature labels this relationship as "BizDev" (Fitzgerald & Stol, 2017). Existing research on information systems (IS), management, and software engineering calls for further investigation of this phenomenon (Fitzgerald & Stol, 2014, 2017). Continuous planning enables business and IT to work closely together in a "BizDevOps" environment (Fitzgerald & Stol, 2014; Silverthorne, 2017). For enhancing the relations of business managers and IT employees, further research is necessary. The aim of this research is to determine *how an understanding of the relationship between customer demands and the DevOps approach can be achieved to enhance continuous innovations.*

We begin with a short introduction of the related literature and present the concepts of BizDevOps, enterprise agility, and continuous innovation. Then, we present our research method and describe our research approach. With the help of the case analysis, we present rich descriptions (Wiesche et al., 2017) of the results. Finally, we discuss our findings and conclude the paper.

5.2 Related Literature

5.2.1 The DevOps and BizDevOps Concept

To achieve a higher success rate of software development projects, a team should integrate skills and broad knowledge about the complete SDLC (Peppard, 2018; Wiedemann & Wiesche, 2018a). For this, the DevOps approach could be a suitable solution, because project team members are responsible for the complete SDLC from planning to operations (Horlach et al., 2017; Wiedemann & Wiesche, 2018b). DevOps is a new technological trend that presents new challenges for organizations (Sebastian et al., 2017).

Business strategy and planning tasks provide challenges for the collaboration of business and IT. According to existing literature, the term BizDev implies the necessity for continuous integration and improvement between business strategy and software development. Hence, BizDev complements the DevOps concept (Debois, 2011; Fitzgerald & Stol, 2017). The importance of closer collaboration between business and IT arises from the short cycles of feedback from customers, which are implemented with the help of agile project management methods (Lassak, Przybilla, Wiesche, & Krcmar, 2017; Przybilla et al., 2018). Furthermore, more and more business employees act as proxies in the role of agile coaches or product owners (PO) in IT projects. To meet and satisfy customers demands, it is essential for the software engineering flow to have a tight connection between business, development and operations (Fitzgerald & Stol, 2017). Continuous planning is a major capability for managing systems (Fitzgerald & Stol, 2014), as done by DevOps teams. Hence, combining BizDev and DevOps to form BizDevOps will foster the collaboration between business and IT. Integrating professional experts into DevOps teams is a key to achieving BizDevOps.

BizDevOps is defined as the integration of domain experts within DevOps teams. A major advantage is that the tighter connection between planning and execution leads to continuous planning (Fitzgerald & Stol, 2017). Hence, customer demands can be satisfied faster and the team can react quickly to changing environments.

Organizations have to rapidly adapt to react to new customer demands (Sambamurthy et al., 2003) and build DevOps and BizDevOps capabilities in order to stay competitive (Sebastian et al., 2017). The reasons are higher customer satisfaction with the provided software, as well as better software quality and higher project success (Tripp et al., 2016). A tighter collaboration between development and the operations part of an IT function is necessary to ensure that errors are quickly fixed and the quality and resilience of the software are enhanced. Nowadays, it is essential to develop innovative capabilities to react to digital disruptions (Châlons & Dufft, 2017).

In traditional IT functions, business managers are responsible for planning and prioritizing the processes. Furthermore, organizations centralize highly specialized IT staff in so-called silo units in order to build new software features using sequential development methods like "waterfall development." Afterwards, there is a long time before the software features are implemented and run by the operations IT unit. The complete process has strong dependencies on the business manager (Brown & Ross, 1996; Markus & Keil, 1994). Through the DevOps concept, solutions are delivered to avoid interruptions at different stages of planning, building, and running. Since the SDLC includes the steps these tasks, a tighter collaboration between planning, executing, and operating is enabled (Fitzgerald & Stol, 2014).

Using the DevOps concept, organizations are able to release new software features frequently and automatically (Ross et al., 2016). Hence, risks linked to software releases can be reduced and feedback for new software features is received faster (Lwakatare et al., 2016).

5.2.2 Enterprise Agility and Continuous Innovation

IS literature provides broad knowledge about enterprise and IT agility but lacks understanding of how a closer connection and flow between business and development and between business and operations can be achieved (Overby et al., 2006). In times of uncertainty regarding planning processes in short cycle developments, agility concepts are necessary. A suitable use of IT is a key leverage factor for organizational agility (Baskerville & Pries-Heje, 2004). Enterprise agility is defined as *"the ability of firms to sense environmental change and respond readily"* (Overby et al. (2006, p. 121) p. 121). Literature highlights that a network based on trust and commitment with blurred boundaries is essential for a relationship between business and IT. A competitive advantage can be gained through better coordination, management, and structuring of relationships with stakeholders and a more agile-oriented collaboration with customers (Christopher & Towill, 2000). Agile software development methods could help to enhance this relationship. IT projects with short time system development enable faster delivery of innovations to the customers (Baskerville & Pries-Heje, 2004). Existing literature states that the combining of business and technology alignment can be achieved and can supports the business cycle, deliver major benefits, and provide innovations (Holmqvist & Pessi, 2006).

Continuous innovation is defined as a sustainable process that supports responsiveness to new requirements and changing market demands throughout the SDLC (Fitzgerald & Stol, 2017; Ries, 2011). In the business context, innovations are combined with new ideas, which are transformed to achieve value for business. Continuous innovation is most widely used in the area of software development through concepts like DevOps. Thereby, early customer feedback to new software deployments can be obtained (Fitzgerald & Stol, 2017). Furthermore, planning is a key prerequisite for continuous innovation. Adequate planning processes are very important for avoiding failures in the development processes. Continuous innovation helps processes to react to new market demands across the entire SDLC of planning, building, and running software (Fitzgerald & Stol, 2014, 2017). The BizDev approach recognizes this issue and tries to tighten the relationships between business strategies and software development. The PO is responsible for the business contact. This is emphasized by agile software development methods like scrum as the first step toward the BizDev direction.

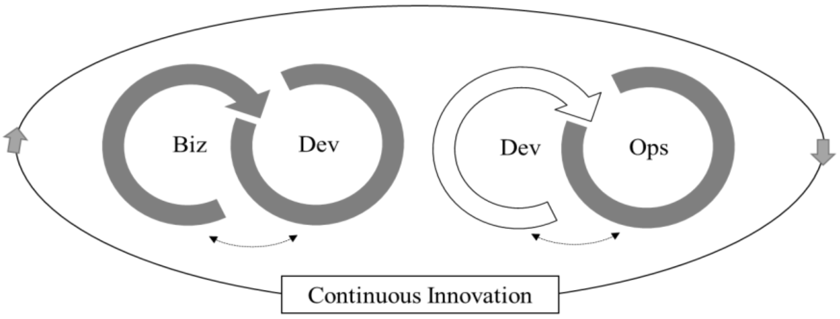


Figure 8. BizDevOps and Continuous Innovation (Own Depiction)

5.3 Research Design

We conducted a multiple-case study to analyze the flow between business stakeholder demands, software developers, and operations. Since BizDev and DevOps studies are neglected in existing literature, our aim is to provide rich descriptions with the help of grounded theory through a multiple-case study research (Urquhart et al., 2010; Wiesche et al., 2017). In this section, we describe our exploratory research design and approach.

The cases considered in our study have their headquarters in Germany. A case study approach is defined as *"an empirical inquiry that investigates a contemporary phenomenon in depth and within its real-life context*" (Yin (2014, p. 18). The present paper is among the first studies to investigate the BizDevOps phenomenon (Yin, 2014). The advantage of case study research is that it can examine real-life situations and test or develop theoretical perspectives in relation to the considered phenomena as they unfold in practice (Flyvbjerg, 2006). In summary, case studies are an appropriate method to improve our understanding of BizDevOps teams and to show how relationships between planning and development and between operations processes of the SDLC are implemented.

Primary Data		
No. of Interviews	**Role of Interviewee**	**No. of Interviews per Role**
23 (some interviews were held with more than one interviewee)	CTO	1
	IT Manager	15
	Product Owner	3
	Team Member	9
	Sum	28
Over 400 pages of transcriptions	The interviewees were mainly conducted personally through face-to-face interviews. Some interviews were held via telephone. The research team	

	took notes regarding observations during the interviews. In total, more than 400 pages of transcriptions and memos per interview were created between the end of 2016 and February 2018.
Secondary Data	
Webpages, blog articles, and white papers	We searched through the internet and collected information about the companies. Often, the companies have blogs where they publish information about collaboration.

Table 26. Primary and Secondary Data (Own Depiction)

We conducted an exploratory case study to answer our research question. Case studies offer a great variety of techniques for data collection (Yin, 2014) for the DevOps teams. Their characteristics and effects on the firms are primarily studied through expert interviews. An expert is someone who has privileged and deep knowledge about a special topic (Bogner, Littig, & Menz, 2009). Here, the experts have privileged knowledge about DevOps teams and the customer view in their organization and can assess their characteristics and outcomes. We tried to talk to people in different roles and responsibilities to find out how planning processes and customer view (Biz) are implemented and how the DevOps concept fosters continuous innovation.

During our research, we used qualitative data-coding processes for the interpretation of our data (Glaser & Strauss, 1998; Urquhart et al., 2010). Furthermore, we followed the guidelines of grounded theory for data collection and analysis, as described by Wiesche et al. (2017).

To participate in our study, a precondition was that the teams had to be familiar with the DevOps concept and must already have integrated planning, development, and operations processes. Additionally, the teams should have integrated an agile method (e.g., Scrum or Kanban) to collaborate with customers. We conducted a multiple-case study to analyze the relationships of business strategies with software development and operations. In short, we talked to 28 interview partners from 15 companies. Table 26 provides information about primary and secondary data.

A semi-structured interview was conducted with each participant, supported by guidelines and a list of questions or general topics that the interviewers wanted to touch upon (Yin, 2014). The questions were mainly open-ended, giving the interviewees the possibility to explore their experience and views (Yin, 2014).

The interview guidelines helped to keep the interaction focused as data collection proceeded. It ensured comparability of data across individuals, settings, and researchers (Maxwell, 2012). Although the interview process was systematic and comprehensive, the interviewer had a high degree of freedom to probe and explore these guidelines. Thus, questions were adjusted during the interviews to gain more in-depth knowledge for each case.

Each interview lasted about 45–75 minutes and was conducted through face-to-face meetings or by telephone. The interviews were held in German or English. German statements were translated into English for further analysis. Every interview was recorded and transcribed. Moreover, a lot of notes were taken during the interview.

During our data analysis, we wanted to examine the relationships and concepts between business planning and software development (Fitzgerald & Stol, 2017). The related literature presented in Chapter 2 was helpful for guiding our examination. For the coding process, we followed the guidelines approach presented by existing literature (Glaser & Strauss, 1998; Wiesche et al., 2017) and used the software NVivo10. During the coding, the research team took notes to justify the coding process. Afterwards, we identified subcategories in the planning and collaboration processes with the help of axial coding. Finally, with the help of selective coding, we related the categories to mechanisms describing the effect of collaboration between BizDev and DevOps.

5.4 Findings

We started with an open coding process to identify core categories of relationships between BizDev and DevOps. The categories explain the process of planning in software development projects and different forms of customer integration into the SDLC. Table 27 presents our findings regarding the open-coding process. These findings confirm the results of existing research about integrating the customer view in software development projects with the help of a PO (Pries-Heje & Pries-Heje, 2011).

Dimension	Definition	Statements (examples)
Team side	The responsible person for planning the backlog, integrated within the DevOps team.	*“From the developer's point of view, I'm very happy that the PO now sits next to me and I can have a constant exchange with him” (Team member).* *“We have a lot of cross-functional teams. That means you have a team with product manager, developers, and QA” (CTO).*
Customer side	The responsible person for planning, integrated at the customer/ business side	*“We work together relatively closely with PMs from other company parts, although they do not sit with us” (IT manager).* *„They are also [organizationally] close to us in the holding company. [...] they call themselves business class” (Team member).*
Team lead	The responsible person of the planning process.	*“I'm in the team in some roles, i.e., in this product team as Product Owner. At first, I also did a lot of development by myself, but everything else the teams do, they just have to vote with the customer” (PO/CTO).*

Table 27. Coding Process and Core Categories (Own Depiction)

5.4.1 Areas of Planning as Subcategories

We present different ways to integrate the customer perspective in the team. On examining the three possibilities of collaboration between business planning and software development and operations, we realized that planning is related to different processes in the SDLC and is connected to BizDev and DevOps activities.

Our findings confirm that the integration of a PO within a DevOps setting is important for achieving continuous innovation. The collected data show that apart from the establishment of the PO in the DevOps team, there are different areas of planning, as described in Table 28. Our interviewees stated that the responsibility for the tasks of planning, developing, and running the software is now integrated into one cross-functional IT project team.

"I know that the Product Owners, who came to us, were already relatively IT-savvy and partly from IT. They have taken a different career path and spent some time in marketing. […] That brought a lot of responsibility into the team" (Team member).

Area of planning	**Definition**	**Code selection**
Responsibility	Planning responsibility means that the team is now responsible and incorporated within the planning process and includes the impact of development and operations.	Writing requirements Service responsibility Requirements implementation Common understanding
Scope	Planning scope is defined as the size and extent of planned components that should be implemented by the team into the software in short iterations.	Agile development meetings Communication Understanding of the software components and customer needs Small increments
Dependency	Planning dependency is defined as the relationship of planning processes that are now integrated within the team with project success and running the software successfully.	Collaboration between planning, building, and running Team autonomy/flat hierarchies Respect among team members Consequences of failures Shared tasks and understanding

Table 28. Areas of Planning Relevant to DevOps Teams (Own Depiction)

Furthermore, the scope of planning changed to a high degree. In traditional software development projects, the customers only have the possibility to plan their requirements for very long release cycles. Hence, business people cannot introduce a new demand into the development cycle because the planning phase is already closed and they have to wait a long time for adding new demands to the next big release (Fitzgerald & Stol, 2017). This problem can be avoided through the implementation of continuous planning with the help of the BizDevOps approach, because introduction of new ideas and requirements is possible at all times.

"So, what distinguishes our team is mainly the Product Owner. A good planning and coordination with the Product Line Management specify the requirements. A really good planning with user stories and not just reacting to requirements but working proactively. […]

We have always really attempted to show a minimal product finished in two weeks—a small increment that we were able to present” (IT manager).

Finally, we identified the area of planning dependencies as related to different processes in DevOps teams. Through the integration of a high degree of autonomy, the team is responsible for the planning process, as mentioned before, and must also be aware of the dependencies for the project success. Since the planning processes are now integrated in the team, the successful software delivery is in the hands of the BizDevOps team.

“Ultimately, the responsibility for the applications lies within the team, and that means—now that the Product Owner is also in the team—actually everything from writing the requirements to development to operation” (Team member).

We listed a row of areas of planning for BizDevOps teams. The introduction of these terms is necessary for describing the process of achieving continuous innovation in software development projects. These terms are dependent on the three core categories identified in Table 28. DevOps teams and business have already implemented planning configurations for achieving continuous innovations.

5.4.2 Mechanisms for Continuous Innovation

As mentioned before, a key prerequisite for continuous innovation is planning. Fitzgerald and Stol (2017) state that innovation in business areas is connected to business value for the service recipient. Continuous innovation tries to enable processes that help to react to new market conditions and are related to metrics across the SDLC. Table 29 presents the mechanisms for meeting continuous innovations related to planning processes, identified with the help of our data. The following table depicts the mechanisms and the related area of planning with the key challenges that appear in our data.

For achieving continuous innovation, our cases initially implemented some mechanisms, which we identified as scalability, security, and quality. Innovation could be gained through scalable services. Scalability fosters speed to easily broaden the resources. The BizDevOps team is now able to do a lot of tasks by itself, because of responsibility for example, and hence to enhance scalability and speed of the service.

“Suddenly, we wanted to scale and that was not so easy in the old structure. With the scaling comes the fact that you want to bring things faster to the customers [...] e.g., during Christmas time” (Team member).

The data presents insights that combine planning processes in DevOps teams; a higher level of unique selling points can be achieved through planning. The team is able to plan its demands and efforts to be taken in case of problems; thus, the team has a great overview.

“Where we want to distinguish ourselves from competitors, which means where we have a higher level of competition there. We also want to have a higher level of agility, in other words, we specifically selected this DevOps-oriented approach because if we want the highest possible speed, then the team should be cross-functional” (CTO).

However, to achieve scalability, a tight exchange of planned requirements is necessary. Furthermore, one interview partner mentioned that they still lost speed because the business department wanted a manual acceptance test: *"Hence, we have to wait for implementation, and I am angry about that*" (Team member).

We identified security to foster continuous innovations as the next mechanism. This is related to the responsibility and scope that are now integrated in the teams. Planning responsibility delivers a feeling of safety to the team; the team is no longer concerned that new requirements would be introduced by external people because they are involved in the planning processes.

"Teams have a higher flexibility; they are more autonomous. They have a higher degree of safety regarding planning" (IT manager).

Security is important during the entire SDLC, and some team leads still make great efforts to *"claim operations responsibility"* (IT manager), because possible failures in the running systems need planning efforts as well.

Mechanisms	Definition	Manifestation
Scalability	BizDevOps teams want to achieve the highest agility and speed to stand out from competitors through integrated planning and solving problems during run time operations and scaling the software to a broader level if necessary.	• Planning of necessary proportional increase of service resources • Claim against competitors • Enhancement of speed through fast reactions and planning autonomy
Related to	Planning responsibility, scope, and dependencies	Need for quicker communication Avoiding of manual acceptance test by PO
Security	BizDevOps teams are responsible for planning and hence want to achieve high planning security, because the team defines the priority of the planned increments of the software if failures appear in the running software.	• Scope and autonomy lead to planning security • Responsibility for services and consideration in planning processes
Related to	Planning responsibility	Need for communication within the team Call for taking over complete service responsibility
Quality	BizDevOps teams are responsible for delivering and running the software. The team members need to be aware that they are responsible if problems appear. Proactive avoidance of failures is enhanced through a high-quality planning processes for the product.	• Product quality depends on planning, developing, and running tasks • Team members develop awareness of product quality
Related to	Planning scope and dependencies	Need for accurate planning of the backlog Call for awareness of failures

Table 29. Mechanisms for Continuous Innovation and Planning Relation (Own Depiction)

The third mechanism that we identified is quality. Our findings indicate that a stable running software, where changes are possible when necessary, fosters innovations. A high-quality planning process avoids failures in the implementation and running phase of the software. BizDevOps teams need the skills and awareness to deliver the complete SDLC.

"The team and its members have very a high level of personal maturity. They have high claims to themselves and to the quality of work. They have a very high degree of customer-oriented thinking" (IT manager).

However, for achieving high stability for a system, planning quality and dependent factors should be considered. *"We want the team to work self-responsibly and decide when things go live and they have to take over the complete responsibility for quality and operations"* (CTO).

5.5 Discussion

In our study, we present rich descriptions of the combination of planning areas and mechanisms for achieving continuous innovations through BizDevOps teams, as presented in Figure 9. We identified areas in BizDevOps and planning processes that trigger mechanisms for achieving innovations for customers, as shown in Tables 28 and 29.

We give insights into how the phenomenon of BizDevOps can be arranged with the related planning processes to understand how continuous innovation can be achieved. This is different from the findings of existing literature, where the need for further investigation of continuous planning and innovation and DevOps capabilities is highlighted (Fitzgerald & Stol, 2017; Sebastian et al., 2017). Thus, the grounded theory approach of achieving continuous innovation through DevOps teams developed in this paper is the first attempt to explain of how planning processes are integrated and how the BizDevOps concept can be achieved.

We propose that this continuous innovation is correlated with the planning of customer requirements. Furthermore, we find that the evolution of continuous innovation is a process that is combined with planning areas. There are some challenges described by our interviewees that need to be considered.

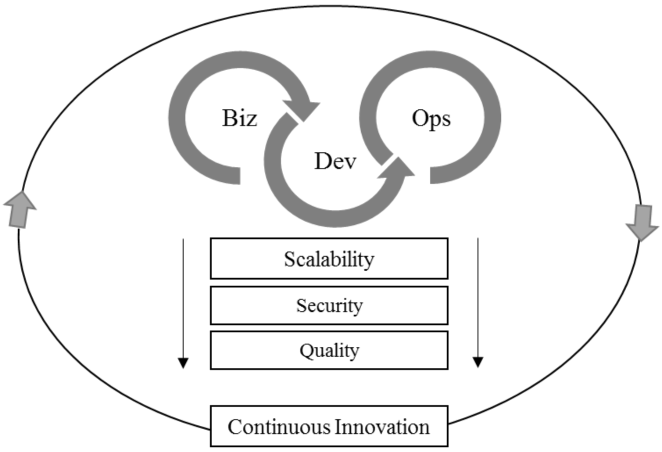


Figure 9. Rich Descriptions of Achieving Continuous Innovation through BizDevOps (Own Depiction)

The first mechanism is scalability. Scalability depends on the planning responsibility of BizDevOps teams. The team members have to feel responsible for the service. If the scaling of the service is necessary due to some peaks, it must be recognized proactively by the team members and accurately planned and realized. Hence, it involves planning responsibility and the corresponding steps. Furthermore, to achieve the necessary speed for scaling, planning dependencies are important as well. The awareness of collaboration and the dependencies of scaling for project success are important points that should be considered.

The second mechanism is security. Security is related to planning responsibility. The BizDevOps team is responsible for planning and operating the system. Hence, there is a need for accurate planning and the awareness must be fostered in the team.

The third mechanism is quality. Quality is dependent on planning scope and dependencies. Within BizDevOps teams, a new culture of collaboration is necessary. The team members need to have the attitude that the service is owned by them. The members have to decide which components are to be developed in each iteration. Therefore, high quality in the planning process is necessary, so that less failures appears.

Additionally, we identified three dimensions of integrating the customer view (Biz) into the DevOps team. Our data indicate that the PO is in the DevOps team, on the customer side, or the team lead. We found evidence that if the PO is integrated within the DevOps team, the best collaboration and planning processes are achieved. However, our findings also indicate that if the PO is settled on the business side, a high degree of exchange and close collaboration with the business are achieved. The third dimension is when the PO is the team lead. This setting appears because of the transformation of traditional IT functions to a cross-functional BizDevOps team setting. Our data highlight that middle management positions could break away, leaving the company with a *"social responsibility"* (Team lead) against managers and the PO position is handed over to them.

5.6 Limitations and Future Research

As mentioned in Chapter 5.3, we mainly talked to IT managers and other IT people. This is because some of our cases lack even a typical business department. Two cases mainly consist of IT departments and some support units. Further research should include cases that have a traditional business department which is involved in the planning processes. Generalization of this study is limited by a case-study approach; hence, validity is limited to our findings. Other studies in different settings might complement our examination. We present insights into how the planning process could be integrated in a DevOps teams with BizDevOps. We present mechanisms for continuous innovation and planning relation, but this needs further enhancements. One way to achieve some kind of evaluation is by conducting a quantitative study. Further research is needed for the replications of our findings across other settings (Yin, 2014). Additionally, future examination is necessary to identify other continuous innovations mechanisms that may be important for BizDevOps settings. Furthermore, we focus on intra-organizational collaboration of customers and IT functions. Other settings with inter-organizational, e.g., outsourcing, settings for development and operations tasks may provide new insights as well.

5.7 Conclusion

Our paper presents insights into continuous innovation steps that could be provided by BizDevOps teams by integrating the planning of customer demands. This is one of the first studies to investigate how planning processes and the customers view can be integrated in a DevOps team—i.e., integrated business processes. With the help of grounded theory, we derived rich descriptions for achieving continuous innovations through the concept of BizDevOps. Thus, we contribute to existing research (Fitzgerald & Stol, 2017; Ross et al., 2016) and provide deeper insights into how the collaboration of business and IT functions could be implemented through new approaches like DevOps teams. Based on our explorative case study, we identified scalability, security, and quality as mechanisms for continuous innovation. Scalability includes the planning processes in case of an increase in service resources, accentuation against competitors, and the enhancement of speed through the possibility of fast reactions. Security in BizDevOps teams refers to the scope and autonomy that leads to planning security and the responsibility for the service and consideration in planning processes. Quality means that the product is dependent on planning, developing, and running tasks that are conducted by the BizDevOps teams and that team members need to develop an awareness of product quality. For practice, we present guidelines for closer collaboration and integration of planning processes and short cycle times in software development projects.

5.8 Acknowledgement

We thank the German Federal Ministry of Education and Research (BMBF) for funding this research (grant code 03FH005PX4).

6. How to Implement Clan Control in DevOps Teams

Title	**How to Implement Clan Control in DevOps Teams**
Authors	Wiedemann, Anna[1] (anna.wiedemann@hs-neu-ulm.de) Wiesche, Manuel[2] (wiesche@tum.de) [1]Neu-Ulm University of Applied Sciences, Center for Research on Service Sciences, Wileystraße 1, 89231 Neu-Ulm, Germany [2]Technische Universität München, Chair for Information Systems, Boltzmannstraße 3, 85748 Garching, Germany
Outlet	Americas Conference on Information Systems, New Orleans, USA, 2018
Status	Accepted
Individual Contribution	Literature research, data collection, data analysis, identification of results, and manuscript writing

Table 30. Fact Sheet Publication P6 (Own Depiction)

Abstract

Clan control is an important factor for project success. A clan is a group with strong social capital. Hence, the implementation of social capital is a precondition for clan control and project success. However, for managers it is still a challenge to build and motivate clan control within IT projects, because people have different skills and backgrounds. Especially in rapid application development projects when new project methods or approaches like agile software development or DevOps are applied. Thus, we build up on existing research and explore how clan control can be build and leverage within cross-functional DevOps teams. With the help of expert interviews with team leads and managers of different DevOps teams, we derived structural, cognitive and relational ties of social capital. Additionally, we present leverage factors from our data for social capital to facilitate clan control and give implications how they can be implemented

6.1 Introduction

More and more organizations organize work in team based structures (Kirsch et al., 2010). Reasons therefore are, that work gets increasingly complex, knowledge intensive and depends on non-routines. With the help of teams, organizations want to react to changing situations (Goh et al., 2013). Teams compose of individual people with different knowledge as well as skills and can act in a collaborative manner to realize a specific aim (Kirsch et al., 2010; Klendauer et al., 2012; Towry, 2003; Wiesche & Krcmar, 2014). IT project success is still an aim that is difficult to achieve.

One possibility to achieve collaboration is control, it *"is defined as any attempt to align individual behaviors with organizational objectives"* (Wiener et al., 2016, p. 742). To achieve a desired behavior, formal as well as informal control can be used (Heumann et al., 2015; Kirsch, 2004; Ouchi, 1978). In context of teams, it is difficult to rely only on formal controls because of individual behaviors and the complexity to measure personal contributions to team success (Kirsch et al., 2010; Schermann, Wiesche, & Krcmar, 2012; Towry, 2003). Information systems (IS) project control is essential to drive or adjust project stakeholders' behavior to motivate them to achieve project aims (Kirsch et al., 2010; Wiener et al., 2016). Recent literature highlighted that there exists a relationship between social capital and informal clan control (Chua et al., 2012; Kirsch et al., 2010; Liu et al., 2015; Wiener et al., 2016).

Clan control can be defined as a type of informal control that appears when the team members behavior is motivated by shared norms and values as well as a strong affiliation feeling to the group with a common goal (Kirsch et al., 2010). Clan control is a phenomenon that appears on team level. Teams' social factors play an important role. Moreover, literature highlighted that social factors are an essential prerequisite of clan control and next to team members, managers play an important role to generate clan control (Kirsch et al., 2010; Towry, 2003). Mangers are often part of the team when they work within or very close with the team (Kirsch et al., 2002; Liu et al., 2015).

In the recent years, the use of agile methods in IT projects gained wide acceptance. It has become a major driver for performance within a lot of IT functions (Cram & Newell, 2016; West et al., 2010). Software development projects remain a major concern for IT managers. Today, companies like Kaiser Permanente apply an approach for teams called DevOps that goes beyond agile (Ross et al., 2016), which is compound of "development" and "operations". This approach helps IT departments to transform to service-centric IT operating models. With the help of DevOps, IT departments are enabled to reduces their software cycle times and bring new features into production in very short time (Ross et al., 2016; Sebastian et al., 2017).

The literature highlight that implementing DevOps capabilities will become a competitive requirement for incumbent companies (Sebastian et al., 2017). Manager are challenged to bring team members closer together to achieve and project success (Chua et al., 2012; Majchrzak et al., 2005). Prior research depicts that efforts are necessary to integrate social capital (Liu et al., 2015) within cross-functional DevOps teams. Social capital is defined as "*the sum of the actual and potential resources embedded within, available through, and derived from the network of relationships possessed by an individual or social unit. Social capital thus comprises both the* *network and the assets that may be mobilized through that" network* (Nahapiet & Ghoshal, 1998, p. 243). Since project members often have different backgrounds from several professions that need to achieve culture of collaboration (Chua et al., 2012; Wiesche & Krcmar, 2014). Within DevOps teams' barriers between development and operations people could appear. There is a need to solved these barriers to gain effective collaboration (Fitzgerald & Stol, 2017). Often complex tasks need to ab addressed by the team, therefore strong collaboration with team members and shared values, understanding as well as social cohesion is necessary (Liu et al., 2015). Hence, we put forth the following research questions: *How can managers build and leverage social capital within DevOps teams to achieve high clan control?*

We argue that managers can promote clan control within team settings for example team-based clan control through the usage of social capital and the implementation of leverage factors that we derived by a qualitative investigation with team leads and IT managers that are responsible for DevOps tams. We investigate how social capital influence the collaboration within DevOps teams. Furthermore, we provide three factors that influence the relationship between social capital and clan control.

6.2 Related Literature

6.2.1 DevOps Teams

Since the presentation of the agile manifesto in 2001, agile software development methods gained high popularity (Tripp et al., 2016; West et al., 2010). A prerequisite of agile development teams is the ability to react on new and unforeseen situations, for example new customer demands, with help of a team (Wiedemann & Weeger, 2017). The aim is to handle the complexity of fast changing environments. Thus, more and more IT functions are implementing product-oriented agile IT teams (Burke et al., 2006; Pflügler et al., 2018; Przybilla et al., 2018). Reasons for integrating agile methods are a higher customer satisfaction with the provided software as well as better software quality and higher project success (Lassak et al., 2017; Maruping et al., 2009; Tripp et al., 2016).

The literature shows that research on agile methods mainly focusses on development activities (Goh et al., 2013; Tripp et al., 2016). IT functions have to enable a tighter collaboration between the different units of development and operations of an IT function to ensure that errors are fast fixed. Hence, the quality and resilience of the software could be enhanced. The literature highlighted, that the IT functions' activities of development and operations have to be a continuous one (Fitzgerald & Stol, 2017). DevOps helps to combine these approaches. For fast providing of new software features, innovations and reacting on problems and failures, IT departments should implement cross-functional teams rather than separated silo IT departments (Fitzgerald & Stol, 2014).

Within many IT organization customers gain deployments of new software code very seldom (Lwakatare et al., 2016). One reason is the poor exchange in form of communication and collaboration between the two department of software development and operations (Fitzgerald & Stol, 2017). For addressing this issues, DevOps provides solutions to avoid interruptions between different stages of the software delivery process (Fitzgerald & Stol, 2014). The software development life-cycle involves the processes of plan, build and run. DevOps helps companies to implement speed and flexibility to deliver rapid development and implement digital innovations when they are needed (Ross et al., 2016). Hence, possible risks through software releases can be reduced, and feedback of a new software deployment faster received (Lwakatare et al., 2016). To summarize, the aim of the DevOps concept is to foster collaboration, automation, virtualization as well as implementing tools for bringing activities of software development and operation closer together (Humble & Molesky, 2011; Lwakatare et al., 2016).

6.2.2 Social Capital and Clan Control

Social capital theory is important for collaboration and addresses the social and cultural preconditions therefore (Riemer & Klein, 2008). The literature presents that managers can support and influence a project by social capital. There exist two forms of control, namely formal and informal (Kirsch et al., 2010; Ouchi, 1979). Formal control focusses on hierarchical power and authority that influence team members to act in a specific way. Informal control relies on self-control through individuals characteristics and/or social relationships (clan control) (Kirsch et al., 2010; Liu et al., 2015). Prior research presents that there exists relationships between social capital and clan control (Chua et al., 2012; Kirsch et al., 2010). A clan is defined as a homogenous group with individuals that share common beliefs, values and norms (Liu et al., 2015; Ouchi, 1979).

Social capital can be measured with help of three dimensions namely: structural, cognitive, and relational dimensions. These dimensions can be implemented as interrelated ties between team members (Nahapiet & Ghoshal, 1998; Wagner et al., 2014). The structural dimension of social capital comprises the contact between the individuals of the network (Nahapiet & Ghoshal, 1998). Structural ties include the settings and forms of communication meetings of the project members which are typically described by frequency, centrality, stability, and density. This could happen physically by colocation, or virtually by emails (Liu et al., 2015; Wagner et al., 2014). The cognitive dimension is defined as the assets that deliver shared understanding and importance among the stakeholders (Nahapiet & Ghoshal, 1998). This dimension includes common language, shared codes, narratives and perspectives as well the project stakeholder's interpretation of reality (Wagner et al., 2014). The relational dimension views on the specific relationships that the people of the relationship have (Nahapiet & Ghoshal, 1998) and includes trust and respect which is a precondition for knowledge sharing. Furthermore, the stakeholders see each other as partners and consult each other for better working together (Wagner et al., 2014).

Social capital is related with networks and relationships between individuals which enhance collaboration. Prior literature treats groups with strong social capital and clans as interchangeably (Chua et al., 2012). Managers can be a part of clans if they work close together with the members (Kirsch et al., 2002). Clan control can be established by the building of social capital. Manager should guide social capital in the clan to reinforce shared values, beliefs and norms that are helpful for project success (Chua et al., 2012). Enacting clan control is bilateral, by building and leveraging the clan. In the first step social capital is developed by structural, cognitive, and relational ties and in the second step, social capital can be leveraged for a specific outcome (Chua et al., 2012; Liu et al., 2015). Within this paper, we derived factors for social capital, afterwards we present how these factors can be leveraged. This research focusses on clan control, because prior research depicts managers could be team members as well and we are interested in building relationships between team and manager (Liu et al., 2015). Hence, we are interested in the development of for example shared norms, values and do not focus on self-control characteristics like self-monitoring of behavior (Wiener et al., 2016) and so on.

6.3 Research Methodology

To answer our research question, we decide to conduct an exploratory research. By using a qualitative research approach, our research can be characterized as interpretive. We want to understand how social capital influences DevOps teams and how they are implemented within existing IT function. Hence, we conduct a row of expert interviews with managers and team leads of DevOps teams. Due to the fact that DevOps is a very unexplored research topic, it calls for case study design (Yin, 2014). The case study approach is defined as *"an empirical inquiry that investigates a contemporary phenomenon in depth and within its real-life context"* (Yin, 2014, p. 18). Case studies are a suitable approach in rare areas in which researcher as well as practitioners search for new insights (Cao et al., 2013; Eisenhardt, 1989). Thus, new findings and a deeper understanding of the topic can be achieved. DevOps settings have to be investigated in their daily environment to gain insights about that approach and how the teams are working with that concept. Hence, qualitative methods are appropriate if the existing body of knowledge lacks of information and questions are open which need further investigations.

6.3.1 Site Selection

For the investigation, the focus laid on DevOps teams. Therefore, we identified contact persons with the, which are engage to the DevOps concept. The focus laid on people, who have experience with DevOps and management position within an IT organization. For the expert interviews, over 50 companies from different industries were contacted via e-mail and telephone, the premise for study participation was that they have already implemented a DevOps team. Our aim was to find cases that have an interesting setting or are easy to replicate to achieve similarities and variation about our findings (Eisenhardt, 1989). In each company, minimum one manager (e.g., team lead) was interviewed regarding their control mechanisms of DevOps teams. The aim was to gain the leadership view on how social capital influence the social relationship with DevOps teams. In summary, we talked to ten managers from seven companies with help of eight interviews (interviews in company 1 and 2 were conducted with both managers at the same time).

ID	Role	Team	Work experience	Industry	DevOps since
I1	Manager	Team 1	10 -15 years	Media	six months
I2	Manager		10-15 years		
I3	CTO	Team 2	5-10 years	Furniture	two years
I4	Manager		5-10 years		
I5	Manager	Team 3	15-20 years	Fashion	three years

I6	Manager	Team 4	10-15 years	Service	five years
I7	Manager	Team 5	10-15 years	Energy	> five years
I8	Manager	Team 6	10-15 years	Health Care	three years
I9	Manager		15-20 years		
I10	CTO/Manager	Team 7	10-15 years	Media	two to three years

Table 31. Characteristics of the Interview Partners (Own Depiction)

As Table 31 depicts, we interviewed managers from different industries. The interviewees had different positions for example CTO or manager. The managers are members of the team and/or responsible for more than one team. The cases have experience with the DevOps approach between six month and five years.

6.3.2 Data Collection and Analysis

The data collection phase took place from October 2016 through November 2017. After identifying possible interviewees, semi-structured interviews were conducted with the participants. Each interview had a duration about 45-75 minutes and was carried out primarily personally through face-to-face meetings or by telephone. In exploratory research personal interviews are recommended because they allow comprehensive discussions. The interviews were held in German or English language. German statements were translated into English for further analysis. The questions were mainly open-ended, that the interviewee had the possibility to explore their experiences and views (Yin, 2014). The questionnaire comprises questions regarding general background of the organization, the use of DevOps principles, and the implementation and enhancement of social capital (e.g., how do the stakeholders communicate?) within the team. The interviews delivered insights into the working style of the teams and informal control mechanisms of the DevOps team. Every interview was recorded.

With the help of the expert interviews with the IT managers, we were able to derive general constructs for social capital as well as factors that leverage clan control. We used coding processes to investigate the relationships between managers and DevOps teams. Coding is very helpful to identify categories and corresponding sub-categories and for examining relationships (Runyan, Huddleston, & Swinney, 2007; Sarker, Xiao, & Beaulieu, 2013). The interview data was coded using software application NVivo 10. We started the coding process following the guidelines of Miles and Huberman (1994). Hence, we started with an open coding process. During the coding, the research team took notes to justify the coding section. Afterwards, the results were analyzed regarding the presented ties of social capital. Then the research team compared their findings for each dimension to identify commonalities, relationships and patterns. The focus lay on the constructs which we identified from literature and the new capabilities that emerged during the data analysis. Furthermore, we analyzed the findings regarding relationships to identify the degree leverage factors within the teams.

6.4 Findings

We analyzed the cases to identify which forms of social capital are established to enhance social relationships through the managers and DevOps teams. We focus on social capital that is built by the managers in each DevOps team to generate ties (Liu et al., 2015; Nahapiet & Ghoshal, 1998). Table 32 provides insights how social capital can be built by managers. These actions are helpful to build effective clan control (Chua et al., 2012).

Construct	Constructs	Example quotes
Structural ties	Enable physical and virtual meetings that support the group.	*“We have eight people in India and two in Germany, [...] they work very close together on a cross-border and daily basis. At the moment we have an overlap of four hours that is very good.”* I8
	Integrate autonomous teams with product responsibility.	*“We are in the process of integrating completely self-organized teams, which means that they are organizationally and technically independent.”* I5
	Implement different forms of meeting structure (team intern and extern).	*“Lectures outside the box so that you look at new tools, new frameworks, new developments.”* I7
Cognitive ties	Spread knowledge within and the team and company.	*“We do not have pockets of knowledge. Of course, we try to avoid that very much. So, it is the worst case, if something like that happens.”* I6
	Establish cross-functional teams with shared knowledge.	*“We do not always have everything to do top down by default we can look how other teams do that.”* I2
Relational ties	Generate awareness of product ownership.	*“That's the part that I'm proud of - we noticed that the teams are going to grow beyond themselves and also bear extra effort to deliver the service in a high quality.”* I4
	Enable freedom within the team.	*“It is nice to see how this freedom develops the team positively.”* I1
	Give responsibility and trust into the team.	*“Management positions will break away and there is a social responsibility towards managers [...] that are 15 years or longer within the company.”* I2

Table 32. Social Capital used in DevOps Teams (Own Depiction)

Table 32 presents activities that helps to build a clan but our interview data presents that distinct actions of managers are necessary to motivate the team after building social ties. This is in line with the findings of Chua et al. (2012) there are two ways of managing a clan proactively. The first one is to reinforce shared norms, values and beliefs that are important for projects and the second one is to constrain beliefs, norms and values that endanger project success. Table 33 presents insights of the purpose and necessary steps that were used by the managers to motivate the team to achieve high clan control. The order presents the frequency and importance of the constructs, for example call for operations responsibility was a major concern of our informants.

Leverage factors	Purpose for managers	Necessary steps
Call for operations responsibility	Team members need to overtake responsibility for operations.	Build the **awareness** for the complete software delivery life-cycle. Team members are equally responsible for development and operations tasks. But the fear of being overstrained should be avoided. If the system is not working, the team is responsible not any another unit in the IT department.
Integrate a culture of feedback and learning	Team members must work within feedback culture and enable continuous learning.	Foster the **willingness** to give constructive feedback and move from a finger-pointing culture to a feedback and learning culture.
Establish receptiveness for fast changes	Team members need to react as soon as possible to changing demands or failures.	Enhance the **attitude** for continuous delivery in fast changing environment. Team members have to learn new actions from development and operations as well as stay up to date to handle new complex changes.

Table 33. Leverage Factors, Purpose and Necessary Steps for Managers (Own Depiction)

Table 33 depicts the norms and values that could be fostered by managers to leverage clan control. Despite a high level of social capital, these factors could help DevOps teams to achieve a high level of clan control.

6.5 Discussion

Our study investigates the implementation of social capital within cross-functional teams. We examined how managers can foster social capital within DevOps teams and what are major challenges for achieving clan control. Now, we want to discuss the major challenges for building and leveraging social capital. Our findings suggest that high clan control is an important factor, because in DevOps settings, team leads move toward a team member position. Nevertheless, our results present that there is still a need for management and guidance of the DevOps team. In the following sections we explain how these relationships can be established.

Our findings indicate that managers have to make great efforts to implement new activities within the teams. For example, integrating operations activities into a development team is a great challenge. Prior research depicts that software operations and development differs in many ways (Edberg et al., 2012). For example developers wanted to implement fast new software features, but operations people want less changes in the system to guarantee stability

(Edberg et al., 2012; Shaft & Vessey, 2006). Hence, development and operations are usually controlled with the help of different modes. These tensions could complicate the building of a clan within projects. We present findings constructs of social capital and clan control how the approximation of different backgrounds could be achieved through DevOps.

Furthermore, our study confirmed existing literature that highlighted that managers can be part of the clan (Liu et al., 2015). If the managers have a high business orientation, the product owner part of an DevOps team might be suitable position for them. Product owners are usually responsible for changes, refine and prioritized the product backlog, which is a list with new tasks that should be built for the system (Schlauderer et al., 2015). I10 demonstrate that he has overtaken the product owner role additionally to his CTO position and the "team together with the product owner overtakes all necessary roles" for managing the service.

Despite being a clan member, for integrating clan control in DevOps settings a suitable management style is necessary. Our findings give insights that the manager role is changing in this flat hierarchies. That means that the manager is seen as leadership person as well as team members. Team lead I6 mentioned that "there are very few situations where I really have to lead and control the team" only in situations of for example major incidents the team leads "phone is ringing first." Additionally, one manager highlighted that "an authoritarian leadership style in the development team is absolutely not working. Laisser-faire always has a bit of an approach that it is not effective. What actually comes out of my experience is an intrinsic motivation" I7. The literature highlighted, control is necessary over several hierarchical levels and that cross-level difference exists in the control styles. Further, there exists different forms of control styles, coercive or enabling (Heumann et al., 2015). Hence, we studied different control mechanisms that are helpful to find out how the control style could be influenced. We highlighted that both forms of control styles are necessary. For example, our findings and leverage factors present that for managing problems, clear guidelines need to be defined, and for integrating culture alignment, enabling styles are suggested. Hence, we summarize that a suitable control style is essential in DevOps teams that fosters intrinsic motivation and create awareness, willingness and attitude for the DevOps approach. We recommend further research in the area of which control styles are suitable for different control situations.

The literature gives insights in how to include social capital into virtual organizations to gain credibility (Riemer & Klein, 2008). Our findings indicate that there is a necessity of structural ties too. Some managers highlighted that they have to organize virtual meetings because they work in offshore (team 6) and nearshore (team 1) team settings. To foster social capital, the interview partners mentioned that they organize regular personal meetings with the near- and offshoring colleagues to share knowledge, communicate and *"there is almost every week either someone from there here or someone from here there. I was there last week and that's very, very important. Especially with new concepts it is really much more effective to spend a week together"* I6. Hence, we summarize that virtual meetings are helpful for the management of existing and stable projects, but if new concepts like DevOps are used it is helpful to meet each team members personally.

Within the area of cognitive ties our findings present that a common and shared understanding as well as knowledge about the tasks of the software delivery life-cycle are indispensable. CTO

I3 mentioned that the *"team is responsible from planning, operations, development, tests, and later for monitoring. The tasks are completely shared in the team."* A high degree of autonomy regarding decision making is typical in agile development projects (Cao et al., 2009). But within DevOps the team has to make decisions about their service that could have influence to the rest of the company. Hence the team must be able to communicate with other teams and managers if decisions *"concerns the system landscape"* I3.

Regarding relational ties, our findings confirm insights from prior research that trust is a major factor for successful team collaboration (Liu et al., 2015). Managers have to trust the team if problems appear *"the team tries to solve it, [if they cannot] an escalation processes will be started [...] this could not be achieved with a command-and-control leadership style"* I4. The *"ownership for the service"* I4 is changing because the complete team is responsible for the service and hence, a feeling of responsibility within in the team can be established.

For a high motivated and aligned team leverage factors are necessary (Chua et al., 2012). In our research we identified three leverage factors that are used by managers to achieve high level clan control: call for operations responsibility; integrate a culture of feedback and learning; establish receptiveness for fast changes. These factors should be focused by the managers when a DevOps team is already established. Managers should have implemented actions to build awareness *"the learning curve should be not too high and [new tasks] should be slowly acquired"* I6. Implementing DevOps approaches need a balance for the call for new responsibility and creating awareness for this necessity. The call for operations is an important leverage factor for managers. Since DevOps describes the development and operations tasks of the software delivery lifecycle, the team members have to overtake new responsibility. I1 mentioned that great efforts are necessary to *"claim for the operations responsibility"* of the team members, because a lot of team members are not willing to overtake this new responsibility.

Prior research shows that tight project schedules are often insufficient for building a clan (Chua et al., 2012). However, our findings present that DevOps teams are often organized in so-called product teams rather than in project teams. That means that they have no clear start or end date and the team members have to work there for an undefined time. Hence, we derive that clan control can be fostered in DevOps settings, because the ongoing team structure presents freedom and time for developing a clan. Additionally, in DevOps team settings it is important to deliver fast feedback to the team since there is usually a high degree of automation (Fitzgerald & Stol, 2017). In project 1 *"these feedback loops were introduced, because we really put the operations pain into the development team and that first hurts and then it gets better"* I2. The team members need to know if something went wrong. Within DevOps settings, the team members should *"work in an open feedback culture and [...] always try to make decisions by consensus"* I10 and work on a common solution. To summarize, the integration of a culture of learning and feedback seemed to be a key factor for leveraging clan control. Furthermore, the receptiveness for fast changes is necessary since team members need to be motivated to work in a DevOps team. Team members need the attitude to steadily work on new solutions and provide it to production in short cycle times (Fitzgerald & Stol, 2017). *"The most important thing is that people fit [in our team]"* I10 from a personal perspective. Therefore, different measures are possible for example the integration of *"an agile coach for the organization"* I3, that helps project teams to learn the DevOps principles and distribute knowledge within the company.

Figure 10 present an overview of the interplay between social capital and leverage factors for clan control.

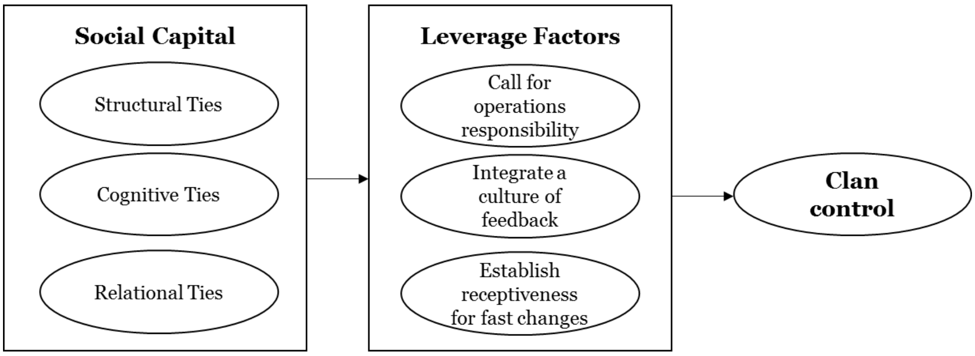

Social Capital
Structural Ties
Cognitive Ties
Relational Ties
Leverage Factors
Call for operations responsibility
Integrate a culture of feedback
Establish receptiveness for fast changes
Clan control

Figure 10. Clan Control as Process of Social Capital and Leverage Factors (Own Depiction)

6.5.1 Implications for Research and Practice

Our findings present insights into how managers can influence the build and motivation of clan control within DevOps teams. The present study enforces the necessity of informal control mechanisms in IT teams (Chua et al., 2012; Liu et al., 2015) and depict how social capabilities can be established and leveraged within DevOps teams. We contribute to existing research and enhanced prior findings of Chua et al. (2012) and Liu et al. (2015). We follow their call for further research in different project characteristics of different organizations and present an investigation of DevOps teams. Management of DevOps teams have an influence to motivate clan control as a controller to build structural, cognitive and relational dimensions of social capital. We deliver observations of social capital that could be facilitated by the managers within DevOps team settings and present a basis for further research. We enhance existing research about clan control in cross-functional IT projects and present DevOps specific social capital and three leverage factors that help to achieve clan control within these teams. Through the present paper an instigation of managing teams regarding their activities of the software delivery lifecycle with focus on software development as well as operations tasks is provided.

For practice we present guidelines for managing DevOps teams from an informal control perspective. We describe the changing role of a management positions to a more coaching and team member-oriented role. Additionally, we identified challenges from our data and subsequently actions for managers how they can handle and motivate clan control. We derived three factors that leverage clan control if they are combined with the ties of social capital.

6.5.2 Limitations

The study provides insights into the clan control of DevOps team, but has some limitations that need to be considered when interpreting the results. The generalizability of the findings is limited, because we only investigated one team and one or two manager perspectives in seven companies, this means that we only examined one DevOps team per organization. In addition, all the organizations are located in Germany. This entails that the applications of the results are limited. Further research could repeat the study in different countries, to examine more project and different project settings per company. Furthermore, a quantitative research could achieve validation for our findings.

6.6 Conclusion

In this paper we demonstrate how managers can support collaboration within IT project teams and build social relationship. With the help of DevOps teams, we examined the influence of social capital for collaboration. Managers should foster structural, cognitive and relational ties through concrete actions. We derived constructs of our data from expert interviews and depict the importance of social capital within project teams. For every tie of social capital, we delivered several constructs that provide insights on how social capital can be built with help of managers to support an IT team. Furthermore, we derived leverage factors for achieving a higher clan control if they are combined with social capital of the team. The three factors are: call for operations responsibility, integrate a culture of feedback and learning as well as establish receptiveness for fast changes. If managers could implement these factors and achieve awareness, willingness, and attitude for the DevOps approach, clan control could be leveraged.

Additionally, we described the changing role of managers within DevOps settings. Managers have to lead the teams on the one hand and act as team members on the other hand. These findings present possibilities for further research for example key characteristics of the leadership style for cross-functional DevOps.

6.7 Acknowledgment

We thank the German Federal Ministry of Education and Research (BMBF) for funding this research (grant code 03FH005PX4). The responsibility for the content lies with the authors.

7. A Control-Alignment Model for Product Orientation in DevOps Teams– A Multinational Case Study

Title	**A Control-Alignment Model for Product Orientation in DevOps Teams–A Multinational Case Study**
Authors	Wiedemann, Anna[1] (anna.wiedemann@hs-neu-ulm.de) Wiesche, Manuel[2] (wiesche@tum.de) Thatcher, Jason[3] (jbthatcher1@culverhouse.ua.edu) Gewald, Heiko[1] (heiko.gewald@hs-neu-ulm.de) [1]Neu-Ulm University of Applied Sciences, Center for Research on Service Sciences, Wileystraße 1, 89231 Neu-Ulm, Germany [2]Technical University of Dortmund, Martin-Schmeißer-Weg 12, 44227 Dortmund, Germany [3]University of Alabama, Information Systems, Statistics and Management Science, Tuscaloosa, AL 35487, United States
Outlet	International Conference on Information Systems, Munich, Germany, 2019
Status	Accepted
Individual Contribution	Literature research, data collection, data analysis, identification of results, and manuscript writing

Table 34. Fact Sheet Publication P7 (Own Depiction)

Abstract

Changes in IT organization and technology environments make it necessary to adapt and review how mission critical IT functions, such as software development and maintenance, align with firm strategy. IT functions increasingly use autonomous cross-functional teams to manage the complete lifecycle of digital solutions. As cross-functional teams begin to alter how we develop and maintain software, they may also result in control-alignment misfits that diminish the efficacy of functional project and operations controls. In this paper, we examine how the integration of product-orientated cross-functional teams challenges and transforms the IT function. To do so, we conducted a global multiple case study of cross-functional DevOps (Development and Operations) teams in large companies. Rooted in control and alignment theory, we present a "bivariate management model" for aligning cross-functional teams and illustrate how shared domain knowledge becomes a critical enabler of product-oriented DevOps teams. This model contributes to IS research through offering insights into how better design control elements of DevOps teams embedded into the IT function.

7.1 Introduction

Advances in information technology (IT) have paved the way for new forms of governance arrangements such as digital platforms (Tiwana et al., 2013), product networks, and for managing digital services (Ross et al., 2016). A key concern for IT functions is to combine flexibility and control in software development and delivery processes (Lamb & Kling, 2003). Control in software development focuses on formal and informal project control mechanisms, whereas control in software operation concentrates on infrastructure and running software applications. Hence, software development and software operations control have different and opposing aims. In response to the specific challenge of aligning control mechanisms for development and operational activities within IT departments (Cram, Brohman, Chan, & Gallupe, 2016b), IT governance has moved from governing functional silo units toward managing cross-functional teams. Creating control capabilities in internal IT teams is necessary to achieve consonant communication and to coordinate autonomous self-managing teams responsible for planning, building, and operating digital solutions (Gregory et al., 2018).

More and more companies are establishing internal cross-functional IT teams to ensure the alignment of information systems (IS) and organizational objectives in IT decisions to fulfill new requirements (Gregory et al., 2018). However, maintaining strong communication and collaboration among employees with different backgrounds has proven elusive for managers (Kirsch, 2004). Thus, communication and integrative collaboration are needed to achieve congruence procedures within the IT department (Onita & Dhaliwal, 2011). Many firms have accelerated this process by adopting software delivery concepts, such as DevOps (development and operations) to coordinate and align disparate communications and opposing goals of development and operations (Krancher et al., 2018).

By combining software developers' and software administrators' perspectives into one self-reliant team (Krancher et al., 2018), firms seek to encourage bidirectional governance and knowledge sharing (Tiwana et al., 2013). There is an inherent conflict of interest between software developers tasked with providing new software code and software operators who require stable software with minimal change (Wiedemann & Wiesche, 2018a). Aligned control mechanisms generate consensus within cross-functional teams and avoid misalignment at an early stage. Extant literature highlights that knowledge is a necessary element of organizational control (Ouchi, 1979; Tiwana & Keil, 2007). Notably, for internal IT units to effectively manage the software delivery lifecycle as self-reliant team, there must be an alignment between the IT knowledge and business knowledge of different departments (e.g., software development and software operations).

This paper aims to provide a better understanding of how firms encourage the alignment of their software development and operations activities. Furthermore, this study seeks to develop a model that explains the interplay between such activities within firms. Existing research focuses on control–alignment in different software development project setups e.g., Cram et al. (2016b)) or on infrastructure governance and control, e.g., Yoo et al. (2010)). Perhaps due to recent developments in relative product-oriented software, few studies have examined how changing control and governance structures affects product orientation in cross-functional IT teams. To fill this gap, we concentrate on the following research questions: *How does the implementation* *of cross-functional teams transform control in development and operations? How does shared domain knowledge influence alignment between software development and operations control elements?*

For answering our research questions, we conducted a multiple case study and used a partial portfolio Grounded Theory approach to analyze the data (Wiesche et al., 2017; Yin, 2018). Conceived in the late 2000s, DevOps facilitates systematic collaboration between software developers and software operations to achieve faster time to market for deploying and operating new software features for digital solutions (Hemon et al., 2018; Wiedemann, Forsgren, Wiesche, Gewald, & Krcmar, 2019). DevOps enables the quick delivery of stable software through continuous delivery, continuous deployment, and solving disagreements between developers and operations through shared responsibility for software delivery processes (Kim et al., 2016). We examine cases in different countries and continents to gain broad insights into cross-functional DevOps teams, focusing of how concepts of control from IS research streams, such as software development and software operations, shed light on managing DevOps teams. Based on our field work, we present evidence of misfits between traditional notions of control and present a model of bivariate management for achieving alignment.

The remainder of this paper is organized as follows. We briefly review the IS control and alignment literature and note relevant research gaps. After describing our research method, we detail our multinational multiple case study and present the results of our analysis of the empirical data. We conclude the paper with a discussion of implications for research and practice.

7.2 Software Development and Software Operations Control in DevOps Teams

DevOps is often considered as an extension of agile software delivery, and also includes operations and maintenance activities (Hemon et al., 2018). DevOps bridges functional silo software development and operations units by integrating cross-functional teams. These teams are responsible for planning, building, and running system processes delivery lifecycles of digital solutions. The integration of cross-functional teams enables faster time to market of new software features through a continuous delivery technology and with higher quality as well as stability through the integration of the operations and software planning phases (Wiedemann et al., 2019). The control mechanisms employed by DevOps teams need to be transformed to enable high performance in a cross-functional working environment (Hemon et al., 2018), but research in this area is rare. Control is a common theme in the software development and IT alignment literature. Control refers to all actions taken to align the behavior of an individual or group with organizational goals (Ouchi, 1979; Wiener et al., 2016). Cram et al. (2016a)) apply this definition to IT, defining IS control as the influence of a person's or group's behavior to achieve objectives regarding the design, development, operation, usage, and management of IS. The software development literature leverages the concept of control to understand how to better design teams and organize projects (Ouchi, 1979; Wiener et al., 2016).

In the following, we review the notions of control found in extant literature and applied to software development and operations. We analyze existing control literature of development and operations, being the two core elements of DevOps. Existing literature explains four control elements: measurement, evaluation, rewards and sanction as well as roles and relationships, that describe the dynamic of control changes during the software development phases (Eisenhardt, 1985; Kirsch, 2004). In addition, we analyze software operations, platform, and infrastructure literature, and mapped our findings from literature onto these control elements (detailed explanations below). These insights from existing research guided our research and theory development for answering the first research question: How does the implementation of cross-functional teams transform control in development and operations? Furthermore, we propose that shared domain knowledge ultimately aligns control between software operations and development in cross-functional teams. Hence, the second open question in this paper is: how does shared domain knowledge influence alignment between software development and operations control elements?

7.2.1 Development Control

At first, we investigated literature in the area of software development control. Because software development is one major part of DevOps, this literature stream guided our research about the transformation of control mechanisms in this area. Extant control literature differentiates between formal and informal modes of IS development projects (Jaworski, 1988; Kirsch, 1996). Formal control focuses on performance evaluation, where either input, behavior, or outcome is measured and rewarded. Informal control concentrates on people and social aspects through clan, and self-control modes (Eisenhardt, 1985; Kirsch, 1996; Mähring, 2002). In controller-controlee relationships, these forms of control are often combined (Ouchi, 1979) and used in portfolios of control mechanisms for supporting management practices in diverse contexts (Kirsch, 1997). This research concentrates on control mechanisms that are used to exercise control. Existing studies of control in IS projects and software development concentrate on four control elements (see Table 35) as a basis for understanding changes in control. These control elements examine the mechanisms that improve the understanding of how control is exercised (Eisenhardt, 1985; Kirsch, 2004).

The first element is measurement that represents specifying and measuring control. Formal measurement presumes that behavior and outcome are explicitly defined and measurable, whereas informal measurement implies that norms, behavior, and values are implicitly defined and measured. Second, the control element of evaluation focuses on mechanisms for assessing performance and information exchange. Formal evaluation builds on specific information relating to behavior, and outcomes, and assesses whether the current situation leads to accelerating progress. Informal evaluation concentrates on norms, behaviors, and values that distinguish a functional relationship through socializing actions like discussions intended to lead to performance. Third, rewards and sanctions as an element based on measurement and evaluation include implicit and explicit gratification and improvement opportunities. The setting is formal when goals or desired behaviors are previously defined and communicated whereas an informal setting is determined whether a behavior is in line with group values and norms. Fourth, the element of roles and relationships is an extension of the three prior elements and refer to an individual's inclusion and relationships. In formal settings, dyadic relationships are in focus, whereas roles in an informal setting typically pertain to relationships among groups of people that depend on each other and committed to common goals (Kirsch, 2004; Ouchi, 1979; Persson et al., 2012).

Element	Formal	Informal
Measurement	• Pre-specified and formally documented goals and/or behaviors are available • Control modes align the goals of controller and controlee • Goals and/or behaviors are measurable	• Few specified behaviors or procedures available • Implicit specification and measurement of group values and norms • Goals evolve over time • Desired end states result when individual behavior is consistent
Evaluation	• Information about rules, procedures, behaviors, and goals are exchanged • Information is exchanged in formal, written documents such as standard operating procedures or status reports • Evaluation assesses whether behavior is resulting in forward progress	• Information about norms, values and expectations exchanged • Socialization, training, discussions, dialogues and meetings serve as mechanisms of information exchange • Goal of evaluation is to build and foster collegial relationships characterized by common values and norms
Rewards and sanctions	• Based on following specified rules or achieving specified targets • Formal organizational mechanisms including pay, bonuses, promotion, or demotion	• Based on acting in a manner that is consistent with group norms and values • Mechanisms include group recognition and peer pressure
Roles and relationships	• Focus is usually on dyads • Controller and controlee are often in a formal superior–subordinate relationship or in a relationship that is consistent with the organizational hierarchy	• Often a work group or professional society • May be a clan, which is a group of individuals who are dependent on each other to accomplish their work and who are committed to achieving group goals

Table 35. Elements of Project Control Adopted from Kirsch (2004, p. 378)

7.2.2 Operations Control

The second literature stream that guided this research is operations control because software operations are the second main part in DevOps teams. These extant investigations present insight of transforming operations control. Including operations tasks in cross-functional teams broadens the agility to software architecture and systems (Hemon et al., 2018). Hence, we propose that integrating operations and development control mechanisms will lead to the alignment of behavior in a cross-functional work environment. Operations control refers to how we shape changes and manage day-to-day activities of the information systems function. IT operations controlling is defined as the monitoring and controlling of IT services and infrastructure. For example, the IT infrastructure library (ITIL), a registered trademark, describes best practice for IT service management processes (Trusson et al., 2014). Operational IT failures of IT systems, e.g., software, networks, depict the presence of IT asset weaknesses considering control mechanisms over IT systems remains important (Benaroch & Chernobai, 2017) because evolving digital infrastructure (Tilson et al., 2010) and platform technologies

(Tiwana et al., 2013) shape the relevance of different control mechanisms. Integrating DevOps teams requires a wide spectrum of operations control mechanisms.

Digital infrastructure underpins contemporary open, heterogeneous, and evolving socio-technical systems and presents challenges for control in participating organizations (Koutsikouri et al., 2018; Tilson et al., 2010). Digital platforms offer services and standards to customers through automated coordination among stakeholders (Benaroch & Chernobai, 2017) and influence management control responsibilities (Gregory et al., 2018). Specifically, research shows software-enabled control mechanisms (Yoo et al., 2010) and platform-based governance mechanisms (Gregory et al., 2018) support autonomous cross-functional teams. We refer to these perspectives as infrastructure control, which is defined as the influence on how users communicate with each other through infrastructure standardization. Infrastructure controls integrates rules embedded in technology as well as norms that shape how users communicate through IT assets within organizations.

System control refers to implementation, changes, and support of software applications. Such application-oriented changes are necessary to meet user demands during releases and after the deployment of new software. Systems control includes monitoring the accomplished tasks of the software (e.g., data flow, logging) for debugging activities and the resolution of emerging problems (e.g., incidents) (Edberg et al., 2012). To better understand how control is changing and enacted in cross-functional teams, we mapped the extant IS operations literature concerning infrastructure and systems control. The four control elements outlined above guided our literature review and we identified key mechanisms and the influence on operational control as summarized in Table 36.

Elements	Infrastructure	System
Measurement	• Pre-defined interfaces for measuring data traffic • Goals are socio-technical (governing collaboration of different stakeholders.) • Measured by input and output parameters and integrated for measuring data traffic from heterogeneous components of an infrastructure (Koutsikouri et al., 2018)	• Pre-defined and formally documented objectives regarding every supporting application (service level agreements/ key performance indicators.) • Goals of software, incident tickets and problems (Nelson et al., 2000) • Speed/cost/value of software releases, ticket resolution
Evaluation	• Information of modular architecture (Yoo et al., 2010) • Information about large-scale infrastructure systems including multiple actors (Aanestad & Jensen, 2011) • Evaluation of whether infrastructure ensures scalability (Constantinides & Barrett, 2014)	• Information about systems/ application • Information exchange via support units/tickets (Nelson et al., 2000) • Evaluation of application stability and outage rate (Edberg et al., 2012)
Rewards and sanctions	• Based on new digital innovation that enable software-enabled infrastructure (Yoo et al., 2012)	• Alignment of actions with specific targets with mechanisms including pressure from other stakeholders, e.g., priorities (Edberg et al., 2012).

Roles and relationships	• Controller and controlee are often at the same organizational hierarchies with a versatile focus (Henfridsson & Bygstad, 2013). • Network of human and non-human (e.g., technology) (Hanseth & Monteiro, 1997).	• Often organized as work groups (support centers.) • Integrated or separate IT unit (Edberg et al., 2012).

Table 36. Elements of Operations Control Adopted from Literature (Own Depiction)

7.2.3 Shared Domain Knowledge as Antecedent for Alignment

Extant literature on the alignment of software development and operations focuses on how the context—specifically, shared domain knowledge—shapes the goals of the IT function with the goals of the broader organization (Carter, Grover, & Thatcher, 2011; Reich & Benbasat, 2000; Tiwana, 2012). Shared domain knowledge is seen as an antecedent of alignment (Reich & Benbasat, 2000). To fully understand what drives changes in control of cross-functional teams, we need to consider shared domain knowledge. Knowledge is considered as shared when persons have knowledge in the domain of an area outside of their own specialized activities and domain (Carter et al., 2011; Tiwana & Keil, 2007). Shared domain knowledge can be symmetrical (Nelson & Cooprider, 1996) or asymmetrical, wherein one individual or unit has greater shared domain knowledge than the other. Some literature examines control and alignment in terms of "peripheral knowledge", which refers to knowledge outside of the specialized domain of activity (Tiwana, 2012; Tiwana & Keil, 2007). Peripheral knowledge contracts the concept of departmental specialization (Ouchi, 1979) because it weakens departmental specialization to achieve other benefits (Tiwana, 2012).

Existing research of shared domain knowledge distinguishes between the firm (IT unit's business domain knowledge and clients' technical knowledge) and project level (e.g., IT units knowledge and partner knowledge in outsourcing arrangement) (Nelson & Cooprider, 1996; Tiwana, 2012). However, research on shared domain knowledge within IT units is rare, which is surprising given that software development and software operations units typically have very different knowledge specialization (Edberg et al., 2012). This suggests a need to further explore shared domain knowledge in software development and delivery because successful cross-functional DevOps teams are likely to require team-level knowledge sharing.

In sum, our research builds on the presented existing literature in the areas of control and alignment. The aim is to understand how shared domain knowledge can accelerate control elements of cross-functional teams, align developers and operations staff, and foster product orientation. This is important because cross-functional teams typically do not operate in a conventional command and control structure (Persson et al., 2012). Because our goal is to derive recommendations of product-oriented control and alignment configurations, we leverage existing research on control balancing (Cram et al., 2016b) and the combination of different portfolios of control modes and elements in project environments (Kirsch, 2004; Kirsch, 1997), focusing on product orientation. Specifically, we see a need to merge control elements from different IS control research streams but also adopt new ones.

7.3 Research Method

To examine control dynamics in cross-functional teams, we apply a multiple case study and used a partial portfolio Grounded Theory approach to understand the setup of cross-functional teams and the control mechanisms necessary to achieve product orientation. This inductive research design allows us to answer "how" and "why" questions about the DevOps phenomenon based on analysis of real-life situations. Therefore, the overall objective of this study is to develop rather than test theory (Birks, Fernandez, Levina, & Nasirin, 2013). As a first step, we analyze the cross-functional product-oriented team phenomenon by applying techniques from Ground Theory Methodology (Birks et al., 2013; Wiesche et al., 2017) and derive the new theoretical model of bivariate management that becomes central to in our theory development.

7.3.1 Research Site

Our multinational case design includes incumbent "brick and mortar" firms in Canada, Germany, USA, Africa, and Australia. The transnational nature of this study arose through respondents to the call for research on control in global contexts due to of its impact on control choices. The international context influences control through differences among time zones, locales, and culture (Persson et al., 2012). The transnational nature of this study was necessary because software development has moved from traditional IT silos toward integrating cross-functional teams like DevOps across the globe; thus, gleaning insight into this movement requires case studies of best practices in firms in different national contexts. Furthermore, we want to identify whether mechanisms and learning possibilities used to spread knowledge and align DevOps teams differ across continents, e.g., Africa, America, Europe, and Australia. Therefore, we searched for examples around the globe that have already implemented DevOps teams to a certain degree.

The firms in our study range in size from 100–250 to > 100,000 employees and are in different industries (see Table 37). All firms use the DevOps method to a significant degree and have internal DevOps teams responsible for a minimum of one IT product. We identified several cases at practitioner conferences at which these companies held presentations and were positioned as outstanding examples or role models for DevOps. We sampled for firms that operated globally, so that we could investigate transboundary learning processes. While all research sites have integrated DevOps-oriented teams, each case constitutes a distinct industry, product line, and customer base. Table 37 presents a brief overview of each participating firm.

Primary data 35 interviews				
Cases and context		**IT department**	**Interviewees**	**Team description**
USA	**CongloCo** A large diversified industrial corporation that is operating worldwide in different sectors and industries, with more than 100,000 employees	IT department is mainly traditionally oriented, with some DevOps teams	Four (Manager, Senior IT Director, two DevOps Engineers)	Seven DevOps team members responsible for frontend service of a portal for customer request for new technology; the team uses Kanban principles
	ComCo A worldwide operating telecommunication provider with a broad range of services and 1,501-5,000 employees	IT function is transforming from traditional to DevOps orientation	Four (Senior Vice President, two Executives, Principal Architect)	Four DevOps team members responsible for cloud-based platform to support other teams; the team uses Kanban principles
Canada	**InsurOrg** One of the leading insurances that offers insurance services in many different areas such as car, home, and life, and provided by 50,001–100,000 employees	High level of traditional orientation and starting to integrate DevOps teams	Three (Assistant Vice President, Senior DevOps Manager, DevOps Manager)	12 DevOps team members responsible for integrating automation processes within the company and other teams; the team uses Scrum principles
	TechCo A technology firm that provides tools for software development and delivery for automation and integration (1oo–250 employees.)	IT function has some internal DevOps teams and has some traditional services	Four (DevOps Team Lead, three Team Members)	Six DevOps team members responsible for developing and managing integration hub for external customers; the team uses Kanban principles
Germany	**GrocCo** A leading food and retail chain company with more than 100,000 employees mainly in Europe; its sales channels are stores and an online shop	The company started transforming some projects to DevOps	Three (Division Manager, Manager and DevOps Team Member, Manager)	Six DevOps team members responsible for configuration management tool and supporting other teams; the team uses Scrum principles
	ServiceOrg An Internet company with more than 1,000 employees; helps end customers identify, compare, and buy products	The complete IT function is managed with the help of DevOps	Five (DevOps Team Lead, two DevOps Team Members, two Site Managers)	Four DevOps team members responsible for product providing web services to end-user; the team uses Kanban principles
	TravelCo A well-known online travel agency with more than 1,000 employees in Germany. Serves customers mainly online	One DevOps team for their platform and some DevOps services	Four (DevOps Team Lead, three DevOps Team Members)	Six DevOps team members responsible for online shop platform; the team uses Scrum principles

Australia	**ResearchOrg** Research consulting institution with more than 250 employees; conducts research in different areas e.g., health	IT function has different teams, e.g., innovation teams and some DevOps teams	Two (DevOps Team Lead, DevOps Engineer)	Nine DevOps team members responsible for the platform and managing network data structures; the team uses Scrum principles
Africa	**FinCo** A leading bank in Africa with more than 50,000 employees started agile transformation and is turning toward DevOps; offers various types of banking products, apps and online services	IT function started as agile and is now turning toward DevOps with the help of the scaled agile framework (SAFE)	Six (DevOps Coach, two Solution Engineers, two Enterprise Architects, Team Lead)	Eleven DevOps team members responsible for digital credit card services; the team uses Scrum principles
Seven Interviews with Experts and Thought Leaders				
• Five prominent DevOps thought leaders who have significantly impacted the practice community through international lectures and publications (all from USA) • One system administrator with over 35 years' experience in a global technology company in USA • One manager with extensive DevOps research and practical expertise in Australia				
>300 pages of documentation	We transcribed all interviews and used memos during the coding process and data analysis, and analyzed company presentations and publications, which fostered discussions with the research team and theory development.			

[1] We used pseudonyms to protect firm identities.

Table 37. Primary Data - Multinational Multiple Case Studies and Expert Interviews (Own Depiction)

7.3.2 Data Collection

We identified leading experts and potential case study participants through targeted conversations and onsite presentations. Our cases include one cross-functional team per interview and several interviews per firm. We conducted semi-structured interviews with open-ended questions face-to-face, by video conferencing, and by telephone (Urquhart et al., 2010). We began each interview with introductions and asking questions about their DevOps-related experience and current relevant responsibility. The main focus of the interview consisted of questions about control and alignment between development and operations, featuring questions such as: What are the benefits of various DevOps team structures? What do you consider as essential components of a DevOps team? What are the similarities and differences between managing IT development and IT operations activities in the team? What are the benefits of various mechanisms to control the DevOps team?

The primary dataset consists of 35 interviews with DevOps team members, DevOps leaders, and senior managers from nine sites in Germany, USA, Canada, Africa, and Australia. After each interview, we documented the insights in analytic memos and the interviews were transcribed. On average, the interviews lasted 54 minutes. To achieve a better understanding of the research phenomenon, we interviewed five additional prominent DevOps thought leaders, a highly experienced system administrator, and a leading scholar and practitioner on the digitization of infrastructure. In sum, we conducted 42 interviews. Using multiple data points helps improve validity and mitigate potential bias (Yin, 2018). To this end, parallel to our interviews, we collected complementary secondary data from, e.g., DevOps conferences, onsite visits to companies, listening to academic talks, and from the practical literature, following scientific guidelines. The Appendix presents information about the secondary data collection and analysis in addition to Table 37. Having multiple investigators in the analysis adds objectivity and enhances creativity to find new contributions (Eisenhardt, 1989). One person conducted the interview and took notes. Afterward the interview's key insights were discussed and summarized within the research team and additional questions for future interviews were formulated (Urquhart et al., 2010).

7.3.3 Data Analysis

We employed a partial portfolio Ground Theory approach for data analysis (Wiesche et al., 2017). We focused on exploring mechanisms used by controllers and controlees of cross-functional DevOps teams and how shared domain knowledge enhances alignment. Furthermore, we examined product-oriented alignment and controlled transformation in organizations. Through constant comparison, we continuously compared the instances of both data and literature to identify categories, thereby contributing to theory development. We used iterative coding and constant comparison techniques to develop open and selective codes through several iterations (Birks et al., 2013; Urquhart et al., 2010; Wiesche et al., 2017). Subsequently, we discussed the findings of coding within the research team to identify the theoretical model (Wiesche et al., 2017). Additionally, we leveraged memo-taking in structuring our coding procedure. We started with open line-by-line coding of all our transcripts based on over 300 interview codes clustered into initial open codes regarding control and alignment. We then identified key patterns through selective coding and linked them to existing literature. These two core concepts **controlled-by-competency** and **controlled-by-invisibility** describe the control concepts of our new theoretical model, which we designate as **bivariate management**. Controlled-by-competency was derived based on the following concepts: leadership transformation, expansion of ownership, and human resource development. The second core concept, controlled-by-invisibility, based on the concepts of supervision, work relief, democratization, and standardization. Detailed descriptions of these categories, codes, and data excerpts are presented in the following section.

7.4 Findings

Our method allowed us to identify control mechanisms implemented by cross-functional teams, which we describe as a "bivariate management" approach. Bivariate management is a set of control mechanisms that enables the alignment of different mindsets, in our case development and operations, by controlled-by-competency and controlled-by-invisibility. Control-by-competency yielded insight into the importance of the human-to-human relationships for control of cross-functional DevOps teams, which requires and results in a high level of competency. In many DevOps teams, members learn and guide each other and evolve into mutual coaches. Control-by-invisibility characterizes control by human-to-technology relationships within cross-functional teams. For example, DevOps teams using software may implement a software-enabling architecture (Yoo et al., 2012) that facilitates controlling of team-related processes. Therefore, the teams use many invisible control mechanisms.

7.4.1 Bivariate Management Enables the Formation of Controlled-by-Competency

In DevOps arrangements, we found that leadership style differ from traditional software development teams. On DevOps teams, team leads or managers need **leadership transformation**, to serve as leaders that can motivate and enable people. *"In terms of the leadership style it is more focusing on the people and figuring out what they need and helping [...], it is more about training and education"* (DevOps Manager, InsurOrg). Moreover, changes required by DevOps leads to a decentralization of responsibilities necessary to bridge development and operations work in one joint team. *"The level of involvement you have to do sometimes to see the way forward. It takes that lead to showing everybody else what we are going to do to make everything better*" (Chief of Staff, ComCo). In transformational leadership style, leaders serve as stewards and flat hierarchies are common forms of DevOps control mechanisms identified by our research.

In DevOps teams, the **broadening of ownership** by overtaking new tasks requires greater competency to master new tasks and capabilities. Organizations give responsibility to their DevOps teams, e.g., for choosing the right technology for their products. This makes the team accountable for the processes of the software delivery lifecycle, e.g., when a team member deploys a new piece of code into a production environment, the team colleagues will be in charge if something is not running smoothly. We recognize accountability as a new control mechanism within the team as well as from the team leadership perspective. One manager said, *"If you have to pay for yourself, you are much more cautious than if you know there is someone else who will fix it for you"* (ServiceOrg). Sometimes, only one person is responsible for managing products or supporting other teams: *"DevOps can either be dedicated people in teams from a specific department or it can be done by individual software engineers"* (DevOps Team Lead, ResearchOrg).

Effective DevOps teams **develop human resources**, by encouraging the learning of new competencies, skills, and knowledge. To better perform essential DevOps tasks, people have to share their expert knowledge in the area of development or operations within the team. *"The main thing has to do with understanding more about the operational space and what that skill set entails to better asses and coach individuals [...]. I don't think that adding operations in development is drastically different; it's simply leveraging a lot of the best practices that we already knew from development from leadership and just expanding those into the operation space"* (Executive Director, ComCo). Bridging development and operations activities within the team requires members to be willing and able to learn new things.

7.4.2 Bivariate Management Leads to the Formation of Controlled-by-Invisibility

All the DevOps teams use automation tools and architecture to organize their work (**work relief**) and minimize the need for manual work. A higher degree of automation leads to fewer manual failures. Developers are responsible for the failures they made while writing a new piece of code. Hence, they want to prevent that failures appear and strengthen their mindsets for higher accuracy of source code. *"We do between four to five thousand changes per month. If they are automated, there's less risk of people making a mistake. The more we automate, the better the availability"* (Vice President, InsuOrg). Consequently, team members rely on automated tools and technology to avoid failures during the deployment of new software features.

While DevOps enables heterogeneous software development methods, our findings indicate the necessity for some **standardization**, as it helps manage huge organizations that have to pass external audits and maintain a high level of security. When necessary, DevOps teams create standards to encourage consistency. For example, one manager of ServiceOrg mentioned *"if anyone says I need ABC or any technology, then my standard question is: what can it that we do not already have […]? I'm really happy if somebody says that, 'we would like to go to ABC because [this tool is no longer supported]', then I say, well, let us think about how we can do it together and then establish it as new standard."* Integrating decision making processes in the team fosters responsibility, but firms restrict the selection of new technology using guidelines to achieve formalization at a negotiable level.

Our key informants explained that using cloud and open infrastructures enables quick software releases of small software code with zero downtime required re-thinking controls. *"You can deploy with zero downtime with the right infrastructure"* (Manager, CongloCo). This requires scaling IT resources to meet varying demands on IT infrastructure. Our informants described how micro services, enabled by container technology or the use of virtual machines from public cloud service providers, offered broader access to developers seeking to commit changes to code. Hence, allowing the DevOps team members to use, e.g., open-source technology calls for the **democratization** of IT access regulations in incumbent firms.

In offering greater access to make changes, DevOps methods resulted in a greater need to monitor and **supervise** changes made by team members. A team member mentioned that *"we check the applications logs to see if the request is handled correctly. So far, we have always been able to identify the problems*" (TravelCo). Key informants reported having to create and implement metrics for measuring and observing application performance, log files, new code, and failures. Many organizations created monitoring technology to continuously monitor processes and changes made by DevOps teams.

7.4.3 Shared Domain Knowledge Enables Bivariate Management

When considering bivariate management, we found that shared domain knowledge enables long-term alignment and provides the base for efficient control-by-competency and control-by-invisibility. More precisely, our findings indicate that the combination of a high level of control-by-competency, control-by-invisibility, and shared domain knowledge is required for achieving high performance in cross-functional teams. DevOps team members' knowledge moves from high specialization toward generalization. *"We usually have to pick up quite a lot of knowledge, and now this [knowledge] is cross-functional"* (Team Lead, ResearchOrg). For controlling DevOps teams, knowledge of both development and operations needs to be shared and adopted by all stakeholders. Our informants described how DevOps teams acquired knowledge from many different sources. For example, they look within their country to digital leaders, e.g., the cases in USA mentioned that they are interested in looking at GAFA (Google, Apple, Facebook, and Amazon) companies (Gregory et al., 2018). Additionally, our interviewees mentioned that they search for knowledge spanning other industries and countries: *"When you see what Toyota did, they did a combination of having folks be really good at those particular tasks but also* *learn faster. So, you had this kind of organizational design stamped out in a lot of places which focused on resource efficiency"* (Senior Vice President, ComCo). Managers also study other cases in the same industry. Case FinCo in Africa investigated the prominent case of ING Diba from The Netherlands and searched for learning opportunities.

The surveyed organizations encouraged learning and acquiring knowledge, investing a lot of effort to educate, train, and socialize DevOps team members to have a learning mindset and readiness to share their knowledge with their peers. One interviewee explained that s/he trains colleagues in the area of Kubernetes, an open-source technology for managing container-applications. *"Like even today we had a little Kubernetes session [...]. A lot of people are interested in learning Kubernetes, and a couple of people on our team, myself and another engineers, know enough to help them and say 'here's what you need to get started'"* (Team Member, TechCo). We found that controlled, aligned DevOps teams have high levels of shared knowledge and product orientation. Because of their deep understanding of the product, the DevOps teams actively engaged in collecting and sharing information with outside teams or open-source communities engaged in producing similar products. *"Now people are you using a lot of open-source packages, right? For example, even if I create a new program, I wouldn't write every piece of code, I would pull in different packages that other people have created"* (Team Member, TechCo).

7.4.4 Misfits in Executing Bivariate Management

Bivariate management is not always performed under ideal conditions resulting in discrepancies in controls and products when stakeholders build workarounds. We refer to these discrepancies as "misfits" in control–alignment product orientation. Misfits in control and alignment arise as a result of local differences in technology availability and also through discrepancies in cultural and local arrangement. For example, comparing FinCo from Africa with cases from USA and Germany, huge differences appear. The availability of technology and infrastructure in Africa is a challenge for FinCo: *"Global research is paramount, and applying best practices in a local context is another. In other words what works in the US and European markets doesn't always work in Africa. You have to take research with a pinch of salt and apply localized parameters and restrictions [...] we have got very unequal society. Some people don't have running water and electricity so how are they going to have a high-speed fiber line? Our solutions have to cut across these kinds of different personas and we have to deliver IT solutions in a very different way because of the type of inequality"* (Enterprise Architect, FinCo). Some public cloud solutions that are standard in USA and Germany are simply not available in Africa. Hence, FinCo builds workarounds and use, e.g., open-source technology and informs themselves on how to manage this problem.

Some identified misfits in control and alignment arose from conflicting approaches of software development in established firm processes. Some of the firms whose employees we interviewed have worked in third-tier support functions for years and made deep investments in ITIL as a means to achieve a high level of stability for running software products. Integrating DevOps into existing IT functions challenged traditional notions of IT service management such as incident and problem management. Companies used to work with ITIL have carefully structured and rigid approaches to IT support because ITIL *"defines [...] processes like:*

strategy design, transition and operations, and those processes are unfortunately the kind of the antithesis of DevOps." However, combining DevOps and ITIL can lead to a *"problem where folks have conflicting goals, right? They have goals of speed versus goals of stability. We really want to align those goals [...and] we had to overcome any problems that ITIL present us"* (Senior Vice President, ComCo). At the time of our case studies, incumbent firms were still working to solve alignment problems by judiciously dissolving some rigid ITIL processes and integrating them into the DevOps teams to enable control.

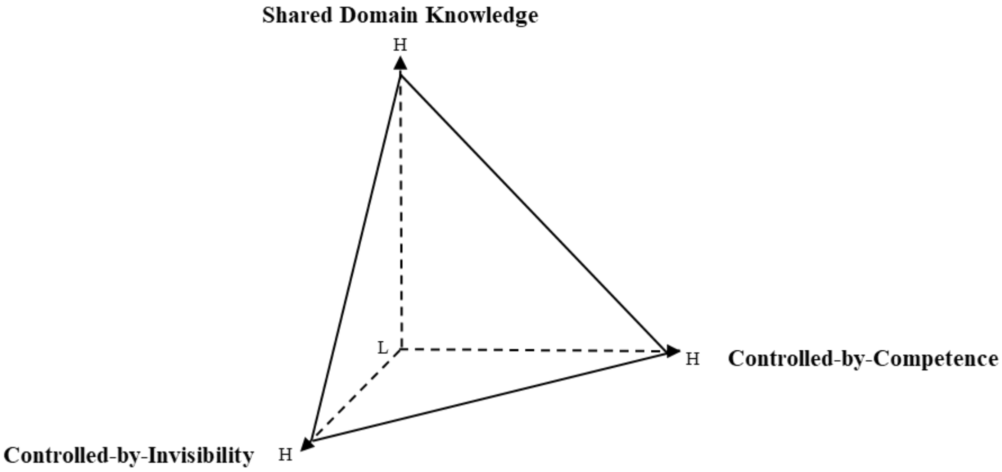
Misfits in control and alignment also arise from large monolithic legacy systems. These systems were often developed to centralize computing and data management. Informants reported that modern DevOps infrastructure teams faced problems, because it is not easy, and sometimes not possible, to transform monolithic systems into micro services. *"API connects with a lot of legacy systems and normalizes the data to be very easy to work with for the portal. We use the gateway in the middle to kind of clean up all the legacy systems and provide an API to the portal. Sometimes the portal suffers [or] some legacy systems suffer maintenance issues or they've just been running for so long and it takes a lot of time to make a small change"* (Team Member, CongloCo).

Misfits in control and alignment also arose from team members' fears about applying a DevOps mindset. Leaders reported fear about changes, such as sharing intellectual property, leading to losing value and power within the firm. Moreover, team members are afraid of sharing knowledge or making mistakes because through open–source technology their specialized knowledge is shared within and outside of the company, thereby limiting their future opportunities. When leaders and managers felt challenged by DevOps, they deviated from bivariate management and reverted to traditional hierarchical structures: *"The biggest problem facing traditional enterprises right now is having the right leaders who really understand technology and the problems and second have the courage to actually make the changes that are required. One, it's hard to find the leaders and second, it's very hard to make these changes because someone has to give up something"* (Senior Vice President, ComCo).

7.5 Control Mechanisms of Bivariate Management

Our bivariate management model has several consequences. First, leaders must effectively align controls from software development and software operations to achieve design controls suitable for DevOps. Second, such control elements are most effective when team members share domain knowledge, which enhances the collaboration of team members with different specialist backgrounds. The results of our study indicate that in both major categories of bivariate management (controlled-by-competency and controlled-by-invisibility), shared knowledge is required to achieve a high level of cross-functional team performance resulting in product orientation. Third, bivariate management solves possible conflict situations at an early stage or before they appear. Merging the different goals and perspectives of team members with different backgrounds leads to achieving a common goal. Figure 11 depicts shared domain knowledge that serves as a catalyst for our identified control mechanisms resulting in a control–alignment tetrahedron. The tetrahedron depicted in the figure constitutes the optimal condition of aligned and controlled bivariate management.

Misalignment appears when control is not exercised simultaneously on a high level. For example, performing control with a great focus on controlled-by-invisibility leads to neglecting the controlled-by-competency part and vice versa. Shared domain knowledge enables the simultaneous execution of control, because it fosters the understanding of the different work that is conducted by team members and managers alike. The tips of the triangle show concurrently high levels of each variable, but can move when misalignment appears.



L=low; H=high

Figure 11. Bivariate Management as Control-Alignment Tetrahedron (Own Depiction)

As the next stage of our study, we present some examples for control elements in product orientation and extend and refine the theoretical framework of control elements presented by Kirsch (2004)). Existing control frameworks tend to focus on either software development or operations control. For controlling cross-functional teams, our data analysis suggested a need to extend or go beyond existing understanding of controls. Our findings suggest that bivariate management needs to combine formal and informal with infrastructure and systems control to accurately describe how controls are enacted for DevOps teams. Table 38 below summarizes the core control elements and lists some examples for DevOps control mechanisms. Instead, of “Roles and Relationships”, we call the fourth element “Relationships and Responsibility” to reflect the broader tasks assigned to DevOps team members.

Elements	Formal and Informal ∞ Infrastructure and System
Measurement	Track benefits and improvement regarding commitments (e.g., learning goals) Enhance agile methods to operations (e.g., operations work on Kanban board) Allow time for unplanned work (e.g., software bugs) Implement metrics for measurement (e.g., user peaks) Set alerts with the help of monitoring (e.g., alters for on-call duty) Automate manual, recurred activities (e.g., autoscaling)
Evaluation	Information about company standards (e.g., obligatory audits) Evaluate value stream management of teams (e.g., essential process steps) Information about allowed tools and technology (e.g., public cloud)

	Information about API, self-service ability (e.g., API to get a new database) Communicate and make agreements within the team (e.g., agree team learnings)
Rewards and sanctions	Implement team milestones and goals (e.g., managing huge tasks) Build scalable, cost-friendly architecture with open-source (e.g., micro services) Resolve failures within the team (e.g., discuss problems in the team) Information about team members tasks and mutual learning (e.g., pair programming) Foster self-reliance (e.g., rewarding through self-motivation)
Relationships and responsibility	Integrate responsibility within team (e.g., managing software product) Make group decisions (e.g., new technology) Responsibility for software delivery lifecycle (e.g., “you build it, you run it”)

Table 38. Control Mechanisms in Cross-Functional Teams (Own Depiction)

7.6 Discussion

The findings from our study shed light on control and alignment in cross-functional DevOps teams and point to a novel bivariate management model. Such a bivariate model is necessary because the practices and expectations of DevOps teams require combining elements of software development and software operations control. Our key informants suggest that effective control of cross-functional teams requires blurring boundaries between principals and agents. This departs from classic controller–controlee relationships described in extant IT controlling literature (Tiwana et al., 2013). One reason for this departure is that cross-functional teams are product-oriented, more focused on outcomes and less focused on process. We found that DevOps teams are not only controlled by a team lead (controller–controlee), but that the team members also control themselves within a team and with the help of technology. To understand how DevOps teams achieve a high level of product orientation, we extend traditional software project control mechanisms and present a new model for managing product teams.

Our bivariate model distinguishes between controlled-by-competency and controlled-by-invisibility. We define controlled-by-competency as human-to-human control mechanisms that are related to human actions with regard to changes. Hence, we expanded on the IS software development project literature and found that formal as well as informal control mechanisms are necessary in DevOps teams. To achieve a high level of coordination among and integration of different autonomous parties requires control changes (Gregory et al., 2018; Tiwana et al., 2013). We identified three major categories of such change: leadership transformation, expansion of ownership, and human resource development. In contrast, we defined controlled-by-invisibility as human-to-technology relationships, including technological aspects used by team members to facilitate their work. These control categories are grounded in software operations literature and we have provided evidence that infrastructure as well as systems control are required for managing DevOps teams. Extant IT control literature focuses largely on project management (Wiener et al., 2016) and overlooks the important role of IT architecture. In distributed organizational arrangements, such as DevOps settings, control embedded in architecture becomes a more subtle and less costly means to control team members’ work. Hence, our findings regarding controlled-by-invisibility are related to previous literature, underscoring architecture as a device effective for coordination and control (Tiwana et al., 2013). Eventually, we found that control mechanisms from development as well as operations are necessary as well as a high level of shared domain knowledge to achieve outperformance of DevOps.

Our model suggests that misfits result in tensions between DevOps team members or in firm processes. At the team member level, lack of shared knowledge creates tensions or inefficiencies. In traditional IT organizations, tensions appear because collaboration and communication between different IT units are rare. For example, software development teams typically have less control and knowledge about infrastructure components (Krancher et al., 2018). Similarly, software operations teams usually lack control and knowledge of coding and software development. This tension results in misunderstandings between employees that limit a firm's ability to implement new software. At the process level, software operations and development processes have different goals (e.g., ITIL = stability; agile = change) requiring different control mechanisms. Figure 11 indicates the optimal approach of control–alignment in product-oriented cross-functional teams. Neglecting one control part of bivariate management will lead to misalignment.

As with all research, this study has some limitations. First, we interviewed mainly DevOps experts from IT. Because cross-functional teams also frequently include business experts, future research should more thoroughly investigate the business perspective. Second, we concentrated on internal DevOps teams, indicating two major categories of controls. However, our interviews also suggest a need to consider other contexts, such as DevOps and outsourcing arrangements, possibly leading to a multivariate theory. Finally, given the qualitative focus of this research, quantitative validation of our insights and model would be valuable, such as a broad questionnaire of diverse DevOps teams across industries on control. Such an analysis is needed to assess the generalizability of our findings.

7.6.1 Implications for Theory

Our in-depth multinational case study of DevOps teams offers several theoretical contributions. Our findings present a novel bivariate management model that underscores the importance of two mechanisms (controlled-by-competency and controlled-by-invisibility) and shared domain knowledge as prerequisite for controlling DevOps teams. First, existing literature depicts self-organizing as a core element of DevOps, which is where no clear role separation exists between developers and system operators (Krancher et al., 2018). However, this view does not explain how to move from a project orientation toward product orientation, which is necessary to successfully deploy DevOps approaches. Our findings underscore the importance of human-to-human control relationships in making this transition because team members learn, and share knowledge with each other, and leaders transform into coaches. Hence, self-organized control hinges on developing product competency and sharing knowledge within the team. The model further develops the agile software development approaches toward agile operations. A high level of alignment between the different control mechanisms requires shared domain knowledge. Specifically, our results show that successful cross-functional teams must achieve a high level of shared domain knowledge to achieve alignment of development and operations control.

Second, existing research of software development control depicts insight into informal control like self- and clan control (Chua et al., 2012; Ouchi, 1979), where different people work toward a common goal or self-monitoring of individuals behavior (Wiener et al., 2016). Research defines a clan as a group with a high level of social capital based on shared values (Chua et al., 2012). This is a very consensus-oriented form enacting control. Within product-oriented teams like DevOps teams, control is based on achieving results and resolving conflict. Developers and operations people have different characterizations and backgrounds, but our bivariate management model lays the groundwork for reducing conflict potential at an early stage by fostering collaborative communications and a results-driven working environment.

Third, our study contributes to infrastructure and platforms literature (Koutsikouri et al., 2018), by providing new insight into the overlooked impact of architecture on control. This paper presents a new form of control for DevOps teams that is embedded in human-to-technology relationships. Cloud platforms and container-oriented approaches enable the agile operations of systems. Our findings present novel insight into how architectural control constrains or enables DevOps team members. We believe that this insight deserves further investigation. For example, as architectures grow more scalable and interoperable, how will different hardware and software configurations shape the way DevOps teams work? How will global technological and architectural changes affect DevOps controlling? How will the opening or closing of digital borders (e.g., the cloud) or regulatory changes such as the General Data Protection Regulation change DevOps members' ability to collect, share, and integrate information? Future research should investigate suitable configurations of control elements in response to unexpected challenges.

Fourth, control elements are mainly discussed in project setups (Kirsch, 2004; Persson et al., 2012). We broaden this framework to include operations control as described in the literature discussion. Moreover, this research presents how control elements regarding “formal and informal” and “infrastructure and system” interact with each other. Managing DevOps arrangements presupposes dynamics in control modes (Tiwana et al., 2013) to explain the movement to product orientation.

7.6.2 Implications to Practice

This work suggests that bivariate management is a useful means for controlling DevOps teams. It also underscores the need for managers to develop new skills, such as coaching and knowledge sharing, to empower DevOps teams. Further, it directs managers to create a culture of support among DevOps team members learning to guide each other, self-manage quality control, and co-define team goals. These skills will help managers improve product management skills in DevOps teams. Furthermore, we illustrate that managers must pay attention to the influence of infrastructure and architectural artifacts, which can both support and limit control and alignment of DevOps teams. Finally, we illustrate that DevOps require breaking down silos typical of software project work. We demonstrate that successful DevOps integrate team members from various backgrounds, create alignment through growing shared domain knowledge, and source/share knowledge with external groups such as open-source communities.

7.7 Conclusion

Our findings introduce a bivariate management model to achieve product orientation, alignment, and control of cross-functional DevOps teams. Our multinational multiple case study makes significant contributions in several areas. We extend existing literature by explaining the controls necessary to effectively integrate software development and operations. Furthermore, we provide a better understanding of the potential of product orientation. In addition, our bivariate management model sheds light on resolving the control–alignment problem and move toward product orientation through shared domain knowledge. The primary outcome of our investigation is a richer understanding of a control–alignment model that helps explain fits and misfits of cross-functional DevOps teams. Our bivariate management model could be used by managers to ensure alignment of product-oriented DevOps teams. We illustrate how existing understanding of controls of software development and software operations can be leveraged to develop new forms of control to manage DevOps teams.

7.8 Acknowledgements

We thank the German Federal Ministry of Education and Research (BMBF) for funding this research under grant code 03FH005PX4.

7.9 Appendix

Exploration and sampling	
Initial exploration	
Selection of suitable case study organizations using theoretical sampling (Eisenhardt, 1989)	Exploration with 10 companies to prepare and develop the present multinational study in order to explore e. g., transboundary learning processes
Scientific conferences and presentations	
Active participation and presentation at scientific conferences in America, Asia, and Europe (e.g., AIS conferences) and research presentation in universities in Germany, USA, and Australia	• Exchange of experiences and knowledge with researchers in the field • Gathering insight about entering the field • Discussion of research ideas
Practical insights	
Expert interviews with thought leaders and industry experts	
Interviews with topic specialists and attending keynotes speeches held by them	Interviews with seven global DevOps thought leaders to gain insights into the novelty of the concept and better understanding practical implementation.
Attending Practice Conference	
Active participation and presentation of research at practice-oriented conferences and	• Exchange of experiences with practitioners and DevOps evangelists as well as knowledge sharing and presentation of prior research results

seminars in USA and Europe (e.g., DevOps Enterprise Summit)	• Identification of participants for second case study

Table 39. Secondary Analysis (Own Depiction)

8. Understanding How DevOps Aligns Development and Operations: A Tripartite Model of Intra-IT Alignment

Title	**Understanding How DevOps Aligns Development and Operations: A Tripartite Model of Intra-IT Alignment**
Authors	Wiedemann, Anna[1] (anna.wiedemann@hs-neu-ulm.de) Wiesche, Manuel[2] (manuel.wiesche@tu-dortmund.de) Gewald, Heiko[1] (heiko.gewald@hs-neu-ulm.de) Krcmar, Helmut[3] (krcmar@tum.de) [1]Neu-Ulm University of Applied Sciences, Center for Research on Service Sciences, Wileystraße 1, 89231 Neu-Ulm, Germany [2]Technical University Dortmund, Martin-Schmeißer-Weg 12 44227 Dortmund, Germany [3]Technical University Munich, Chair for Information Systems Boltzmannstraße 3, 85748 Garching, Germany
Outlet	European Journal of Information Systems
Status	Accepted, published online
Individual Contribution	Literature research, data collection, data analysis, identification of results, and manuscript writing

Table 40. Fact Sheet Publication P8 (Own Depiction)

Abstract

A top priority of organizations around the globe is to achieve IT-business alignment at all levels of the organization. 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 organizations 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.

8.1 Introduction

IT-business alignment remains a key concern among information technology (IT) executives (Gerow, Grover, Thatcher, & Roth, 2014; Reynolds & Yetton, 2015). Traditionally, information systems (IS) functions are divided into separate subunits, including software development and software operations (Hemon, Monnier-Senicourt, & Rowe, 2018). This organizational structure hinders cross-functional collaboration and alignment across different subunits within the IT function (Constantinides & Barrett, 2014; Gregory, Kaganer, Henfridsson, & Ruch, 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, Luther, & Jost, 2018).

Establishing consistency between such organizational functions to realize 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, Beimborn, & Weitzel, 2014).

Recent alignment literature has identified several dimensions of misalignment in the IT function. First, as development cycles grow shorter, the development prioritizes providing new software features quickly, while operations prioritizes ensuring stable running systems with as few changes as possible (Edberg, Ivanova, & Kuechler, 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 organization 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, Cao, & Baskerville, 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, Onita, Poston, & Zhang, 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 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, Humble, & Kim, 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, Wiesche, Gewald, & Krcmar, 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.

8.2 Related Literature

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

8.2.1 Agile Software Development and DevOps

IT functions are typically divided into several subunits, often including separate units for software development and operations (Fitzgerald, Hartnett, & Conboy, 2006; Swanson & Beath, 1989). Traditional approaches to managing software development include top-down planning and sequential implementation (Bick, Spohrer, Hoda, Scheerer, & Heinzl, 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 organizations desire to align software development and operations to facilitate collaboration (Wiedemann, Wiesche, Thatcher, & Gewald, 2019).

To meet business requirements through better software development, more and more organizations are switching from traditional software development approaches to agile software development methods (Bick et al., 2017; Maruping, Venkatesh, & Agarwal, 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, Mithas, Schmidt, & Heinzl, 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).

8.2.2 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 realizing 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, organizational 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.

8.2.3 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 customize IT systems to meet current strategic needs, resulting in an inflexible, substandard infrastructure which is costly to update (Shpilberg, Berez, Puryear, & Shah, 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 summarized in Table 41 below.

From a **goals** perspective, intra-IT development and operations activities can have different scopes and pursue different aims (Fichman & Melville, 2014). Developers prioritize innovation to realize 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 work-flows 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.

	Description	Intra-IT Misalignment
Goals	Align operational IT goals and business value goals (Gerow et al., 2014a; Rivard, Raymond, & Verreault, 2006). Alignment involves achieving commitment by linking objectives across functional areas capabilities. Close alignment positively impacts performance (Bharadwaj, Bharadwaj, & Bendoly, 2007; Powell, 1992).	Misaligned business and IT goals threatens cost efficiency and stability in turbulent environments (Gerow et al., 2014a; 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 organizational views (Edberg et al., 2012; Fichman & Melville, 2014).
Procedures	Align operational IT processes and technology with business processes to benefit customers (Barua, Konana, Whinston, & Yin, 2004). Alignment can be achieved by continuously adapting reconfiguring organizational and IT infrastructure and processes to create business value (Chan & Reich, 2007; Vermerris, Mocker, & van Heck, 2014).	Misaligned intra-IT processes, infrastructure and workflows can threaten customer satisfaction (Kang, Park, & Yang, 2008). Developers use flexible lightweight processes (agile manifest) to achieve innovation and change (Bick, Spohrer, Hoda, Scheerer, & Heinzl, 2017; Tiwana & Konsynski, 2010). Operations favor 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, Wagner, & Beimborn, 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.

Table 41. Perspectives on Operational Alignment and Forms of Intra-IT Misalignment (Own Depiction)

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

8.3 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 analyzing alignment in DevOps teams within multiple organizations, 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 theorizing technique. The generalization 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).

8.3.1 Data Collection

We collected primary and secondary data on DevOps cases from organizations in eight industries in Germany by holding 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 organizations. 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 organizations 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 organizational setting, responsible IT service, organizational 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 e-mail, 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 42 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).

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* 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* 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* interviews with division manager, two managers, and one team member
Case 8	A well-known on-line 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
*** interviews were held with two people**		

Table 42. Case Description (Own Depiction)

8.3.2 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 undertook theoretical sampling, rigorous coding, memo writing, and constant comparison when analyzing 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, Jurisch, Yetton, & Krcmar, 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 conceptualization based on theoretical sensitivity (Glaser, 1978). In line with 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 synthesize 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, Fernandez, Levina, & Nasirin, 2013). We relied on previous research during our data collection phase, recognizing 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 43 presents an overview of our open coding.

Mis-A	Open Codes
Mis-A1: Goals	• Significant effort spent managing old legacy systems • IT function still works with legacy systems managed by data centers • 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 where the problem comes from (own service or other) • Planning backlog without operations or development

Table 43. Misalignment between Development and Operations and Open Codes (Own Depiction)

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 44).

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

Table 44. Overview of Tripartite Intra-IT Alignment (Own Depiction)

Across all our cases, we found various components of our three alignment mechanisms. While each case emphasized 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.

8.4 Findings

Our analysis reveals three core mechanisms used to achieve intra-IT alignment between development and operations in cross-functional teams: individual componentization, integrated responsibility, and multidisciplinary knowledge (see Table 45). Our results suggest that the interplay between these three mechanisms address the different forms of misalignment identified above.

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 authorization through self-services for teams regarding technology selection in order to achieve individually configurable end products.	• Silent releases • Containerization • 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

Table 45. Three Alignment Mechanisms of Intra-IT Alignment (Own Depiction)

8.4.1 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.

8.4.1.1 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, containerization** and **convertible infrastructure**. Before establishing DevOps teams, all of the organizations 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 organizational 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 toward 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 **containerization**. Containerization 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. Containerization 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 organize 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 authorizes 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).

8.4.1.2 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 customization of one or more agile methods to achieve the necessary process guidance and also individualized 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 organized 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 endeavor 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).

8.4.1.3 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 specialize 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 organizing 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 specialized 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 organization 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 recognized 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).

8.5 Discussion

Modern software development ecosystems require high degrees of orchestration (Huber, Kude, & Dibbern, 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.

8.5.1 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 12). 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 organizations 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.

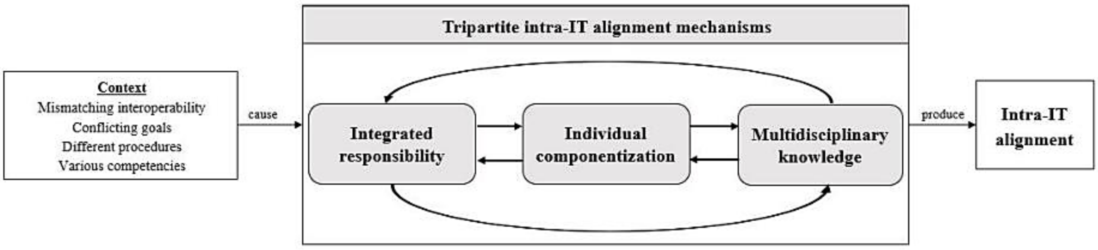

Context
Mismatching interoperability
Conflicting goals
Different procedures
Various competencies
cause
Tripartite intra-IT alignment mechanisms
Integrated responsibility
Individual componentization
Multidisciplinary knowledge
produce
Intra-IT alignment

Figure 12. Tripartite Intra-IT Alignment (Own Depiction)

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.

8.5.2 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, containerization 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 containerization 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.

8.5.3 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).

8.5.4 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 institutionalized 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).

8.5.5 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 containerization 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 organizational goals.

8.5.6 Implications for Practice

Table 46 below provides practical guidelines for aligning development and operations in joint DevOps teams.

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 modernize the IT landscape and support self-organization 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.

Table 46. Practical Guidelines and Recommendation (Own Depiction)

8.5.7 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 organized DevOps teams that are accountable for smaller software products like online shops. More complex software products, sourcing options and inter-organizational 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-organizational relationships—e.g., outsourcing options—and how these are related the domains of SAM.

8.6 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.

8.7 Appendix A

Description of Firms in Study

Case	DevOps Team Setup
Case 1, a leading food and convenience retail company: The company is organized 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 decentralized service support • Scrum principles for tasks organization • 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 organization
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 organization
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 organization with DevOps.	• Mainly development background • Features development, automation monitoring tasks • 24/7 service support and by team lead at night • Kanban principles for tasks organization

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 organization.
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 organization
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 organization • 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 on-line 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 organizes their internal online store platform.	• Mainly operations background • Infrastructure set-up, automation, scripting, system configuration tasks • 24/7 service support • Scrum principles for tasks organization
Secondary data: Elaboration of sketch of the team structure, company presentation, blog articles, and publications	

Table 47: Descriptions of Case Study Participants (Own Depiction)

8.8 Appendix B

Interview Questions (Excerpt)

Personal and organizational demographics

Please introduce yourself (background, education, experience, role, etc.)?

How is your organization structured (organigram, staff, management, etc.)?

Team-related issues

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?

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?

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?

Others:

What mechanisms do you use to align development and operations?

What are major challenges in achieving alignment?

PART B2

PUBLICATIONS UNDER REVIEW[2]

[2] The articles under review have been slightly modified, including the unification of format and reference styles, the correction of spelling errors, and minor grammatical revisions.

9. Combining Development and Operations: Toward a Control Model in Cross-Functional Teams

Title	**Combining Development and Operations: Toward a Control Model in Cross-Functional Teams**
Authors	Wiedemann, Anna[1] (anna.wiedemann@hs-neu-ulm.de) Wiesche, Manuel[2] (manuel.wiesche@tu-dortmund.de) Gewald, Heiko[1] (heiko.gewald@hs-neu-ulm.de) Krcmar, Helmut[3] (krcmar@tum.de) [1]Neu-Ulm University of Applied Sciences, Center for Research on Service Sciences, Wileystraße 1, 89231 Neu-Ulm, Germany [2]Technical University Dortmund, Martin-Schmeißer-Weg 12 44227 Dortmund, Germany [3]Technical University Munich, Chair for Information Systems Boltzmannstraße 3, 85748 Garching, Germany
Outlet	Information and Organization
Status	Major Revisions
Individual Contribution	Literature research, data collection, data analysis, identification of results, and manuscript writing

Table 48. Fact Sheet Publication P9 (Own Depiction)

Abstract

Information systems (IS) literature has predominantly studied IS project control, which concentrates on software development projects. By virtue of digital transformation, organizations start to implement cross-functional teams and combine software development as well as software operations tasks to swiftly react to changing environments. Software operations projects or a combination of control practices relating to software development and operations provide the potential for further research. DevOps (development and operations) recognizes the interaction of development and operations activities within a cross-functional team. This paper investigates the different perspectives for addressing control over the complete software delivery lifecycle. With the help of a multiple case study, six internal DevOps teams were studied. We explain how controller and controlee can collaborate in practice. This paper builds on grounded theory and derives six types of tensions while combining development and operations tasks in one cross-functional team. We depict a DevOps control model that explains the routes that can be taken to successfully address these tensions and enable successful product management within the team. The results reveal that a control model can solve these tensions. Furthermore, combining development and operations control modes to manage cross-functional teams are explained in this paper. Hence, four control modes and their related control configurations are delivered through this research.

9.1 Introduction

Governance and control mechanisms in project management remain a key concern of information technology (IT) executives and CIOs (Cram et al., 2016a; Kirsch et al., 2010). In spite of continuous research effort, literature confirms a need for improving control practices, because project success is often an elusive aim (Liu et al., 2015). Existing inquiries mainly observe either control in software development (termed hereafter as "development") (Dhaliwal et al., 2011; Eseryel & Eseryel, 2013; Maruping et al., 2009) or in software operations (referred to hereafter as "operations") (Banker, Datar, Kemerer, & Zweig, 2002; Bharadwaj, Keil, & Mähring, 2009; Nelson et al., 2000) instead of combining the two activities.

The development team's project success is predominantly measured using time- and budget-related objectives (Choudhury & Sabherwal, 2003; Cram & Newell, 2016). However, while looking at the software delivery lifecycle, the main amount of the budget and work effort is spent on post-implementation processes such as operations (Banker et al., 1991; Banker et al., 1998; Edberg et al., 2012; Stachour & Collier-Brown, 2009). Activities pertaining to operations largely pay attention to the modification of software changes after deployment, resolving performance issues, and adapting the software to changing environments (Banker et al., 2002; Kim & Westin, 1988).

Until now, few studies have examined the control mechanisms for combining development and operations tasks in cross-functional teams (Goh et al., 2013). The goal of this paper is to investigate control modes in teams that combine their development and operations activities to achieve synergetic effects. However, development and operations activities pursue opposing goals. The combining of these two actions through integrating cross-functional teams can lead to tensions (Onita & Dhaliwal, 2011; Shaft & Vessey, 2006). It is yet unclear how these tensions can be resolved (Edberg et al., 2012) as literature rarely presents insights into the aspects of control configurations in cross-functional teams.

In this study, we assess the interplay of development and operations activities which are integrated and controlled in cross-functional teams. We particularly focus on control theory to explain the process of managing teams to achieve task completion (Kirsch, 1997; Maruping et al., 2009). It is largely unclear how the control modes of development and operations can be successfully influenced to manage this syndetic relationship of development and operations. Accordingly, the following research question appears: *How can control mechanisms guide development and operations activities in cross-functional teams?*

To answer this research question, we use a grounded theory approach (Glaser, 1978). We aim to achieve control configurations for managing cross-functional teams. For this purpose, practices like DevOps, the combination of development and operations activities, are established within IT departments (Debois, 2010). The study of DevOps as an empirical phenomenon within IT is necessary for reducing software cycle times through continuous delivery and by enabling fast feedback (Fitzgerald & Stol, 2017; Sebastian et al., 2017). The research institute Gartner Inc. predicts that by 2020, 20 percent of their clients will use DevOps concepts to complement their traditional IT function (Laney & Jain, 2016).

Our findings present six potential tensions between development and operations activities while integrating cross-functional DevOps teams in existing organizations: (1) integrating automation versus manual regulations, (2) initiator versus detector of system failure, (3) claiming more self-responsibility versus centralized hierarchies, (4) generalist versus specialist knowledge, (5) combining different background versus creating a culture of learning, and (6) implementing cross-functional teams versus traditional silo IT. We contribute to the existing literature by building on the control modes of development and operations; we illustrate the combinations of these modes for solving tensions and achieving suitable control configurations in cross-functional DevOps teams. Our results issue new models for the design and implementation of control to help managing these organizational initiatives.

In the following sections, we explain the insights derived from prior research on development and operations control and introduce the concept of DevOps. Afterwards, we describe the research methodology, present our results, and discuss our findings. We conclude by mentioning some limitations of this paper and finally discuss avenues for further research.

9.2 Practice-oriented Background and Theoretical Foundation

In this section, we start with a short overview of the DevOps concept. Moreover, we provide insights into IS project control, which is a key activity in software development (Wiener et al., 2016) and address operations control in the area of IS literature (Cram et al., 2016a). Consequently, we observe the rare existing research that combines development and operations tasks.

9.2.1 The DevOps Concept

DevOps is a highly discussed topic in practice literature and the white papers of well-known companies (e.g., IBM, 2013). Little research has been conducted that addresses this topic and research calls for further examinations (e.g., Fitzgerald & Stol, 2017; Sebastian et al., 2017). The academic and practice literature highlight that a tighter collaboration between the development and the operations part of an IT function is necessary to ensure quick fixes of errors and to enhance the quality and resilience of the software. DevOps can help to combine these activities and reduce software cycles by pushing new codes quickly into the production environment (Fitzgerald and Stol 2017). We define DevOps as a cultural and technological concept for integrating the tasks, knowledge, and skills pertaining to the planning, building, and running of activities into a one cross-functional team that is responsible for one or more IT service products. To rapidly deliver new software features, innovations, quick handling of problems, and integrating operations activities, IT departments should integrate cross-functional teams rather than separate activities in silo architectures (Fitzgerald & Stol, 2014; Nicole Forsgren et al., 2018a).

The aim of DevOps is to foster collaboration, automation, virtualization, and tools to bridge the activities of development and operation (Reed, 2014). Through DevOps, solutions are delivered to avoid interruptions between different stages of the software delivery process (Fitzgerald & Stol, 2014). The entire software delivery lifecycle comprises the following steps: planning, development, and operation tasks. DevOps helps companies to acquire the necessary speed and flexibility to ensure constant and rapid development and implementation of digital innovation (Ross et al., 2016). Hence, risks associated with software releases can be reduced and feedback for new software features is received faster (Fitzgerald & Stol, 2017). Development comprises the analysis of new requirement, design and coding activities, and verification and testing, while operation considers maintenance and software installation tasks (Fitzgerald & Stol, 2014). Traditional silo-oriented IT functions foster the specialized knowledge of one area in one department but DevOps teams need a very broad general setup over all the tasks of the software delivery lifecycle (Rouse, 2016).

9.2.2 Control in IS Software Development

Control in development is defined as the management "attempts to ensure that individuals working on organizational projects act according to an agreed-upon strategy to achieve desired objectives" (Kirsch, 1996, p. 1). To organize projects and to achieve their objectives, some strategies are adopted for controlling the activities of the project members in a functional way (Gregory et al., 2013; Ouchi, 1979; Wiener et al., 2016). Prior literature demonstrates different types and modes of control activities. Ouchi (1979) provided the well-known control modes with behavior and outcome control as well as informal clan control. These control modes were complemented by input control (Jaworski, 1988) and self-control (Henderson & Lee, 1992); they can be summarized into the topics of formal and informal control. Definitions of the control modes are presented in Table 49:

IS Software Development Control

Control Mode	Subgroup	Definition	Reference (examples)
Formal Control	**Input Control**	It is defined as the management of human, material, and financial project resources. The controller wants to specify, monitor, and influence resources for project tasks.	Jaworski (1988); Mähring (2002)
	Behavior Control	It is defined as the influence on processes to achieve the desired goal by stipulating concrete procedures and rules. The controller monitors tasks and the profitable compliance of members.	Heumann et al. (2015) Henderson and Lee (1992); Kirsch et al. (2010)
	Outcome Control	It is defined as the interim and final output provision. The controller determines clear objectives while the controlee is rewarded while achieving these goals.	Harris et al. (2009); Heumann et al. (2015); Maruping et al. (2009)
Informal Control	**Clan Control**	It is defined as the behavior of a group which is motivated by sharing norms, values, and common vision. Clan control is mainly implemented through the controlee; the controller is often outside of the group.	Chua et al. (2012); Kirsch et al. (2002); Mähring (2002)
	Self-Control	It is defined as intrinsic motivation and individual standards as well as the goals of the	Kirsch (1996); Tiwana and Keil (2009)

	controlee. The controlee defines the objectives and the necessities to achieve it.

Table 49. Control in IS Software Development Projects (based on Wiener et al. (2016))

Formal control modes are characterized by concrete prescription from the controller while informal control tries to influence factors of the behavior of controlees (Jaworski, 1988). The controllers build so-called control portfolios for control through the use of several control modes—formal and informal (Choudhury & Sabherwal, 2003; Kirsch, 1997).

Initial research concentrates on internal IS projects (e.g., Henderson & Lee, 1992) but, recently, control theory has paid a lot of attention to outsourced/offshoring IS projects (e.g., Choudhury & Sabherwal, 2003; Gregory et al., 2013) or IS development control in agile development teams (e.g., Maruping et al., 2009). The literature highlights that effective control modes are those that enable autonomy in using the methods of achieving project goals. A major challenge is the management and enabling of the effective team work (Maruping et al., 2009).

Maruping et al. (2009) demonstrates the fact that control modes do not appear in isolation and are not independent from each other. Project managers apply a portfolio of controls to manage IS projects (Choudhury & Sabherwal, 2003; Kirsch, 1997). Control portfolios include both formal and informal control mechanisms (Kirsch, 1997). The literature mentions how the sub-dimensions of control environments, for example, organizational design, influence the way in which organizations exercise control mechanisms (Choudhury & Sabherwal, 2003; Cram et al., 2016a; Kling & Iacono, 1984; Orlikowski & Daniel, 1991). Relevant research points out that inter-personal structures integrated in IS processes influence decisions on control mechanisms (Cram et al., 2016a). Overall, we summarize that inappropriate control mechanisms of development activities in cross-functional teams can jeopardize the success of these collaborations.

9.2.3 Control in IS Software Operations

Prior inquiries address various frameworks for defining, assessing, reporting, and improving internal operations control in organizations (Cater-Steel & Tan, 2005; De Haes & Van Grembergen, 2004). Operations include the definition of software maintenance in accordance with the IEEE 1219 standard as a modification of software toward concrete errors to enhance performance or yet other issues, or to adapt the software in a changing environment; they include activities such as analyzing business processes for deriving new opportunities (IEEE, 1998). Furthermore, operations help users with new or modified demands and assist in terms of technology and platform changes (Edberg et al., 2012; Nelson et al., 2000). The Control Objectives for Information and related Technology (COBIT) provide high-level control objectives and a management approach for 37 IT processes (Cram et al., 2016a; ISACA, 2012). COBIT is an international control framework that addresses the IT governance issues related to project management and provides structure and metrics that are essential for performance measurement and control systems (Melnyk, Stewart, & Swink, 2004). The COBIT control objectives support the IT customer's processes, for example, by establishing a helpdesk. These high-level control objectives can be integrated with the help of the IT Infrastructure Library

(ITIL). Managers use ITIL to structure their operations processes through this best practice framework (Trusson et al., 2014).

Prior research focused on different control mechanisms in operations. For example, Banker et al. (1994) mention release control process while the user requests for modification of software are grouped into package and installed with the help of a release. Furthermore, service support control processes are mentioned in the literature (Nelson et al., 2000). This form of control is helpful for incident management, right from the appearance of the incident to its fixing with the help of a system change or release (Cater-Steel & Tan, 2005). We summarize the current literature on IS operations control and present an overview of the findings in Table 50.

		IS Software Operations	
Control Process	**Subgroup**	**Definition**	**Reference**
Implementation Control	Release Control	It refers to the implementation of user demands that are necessary for software modifications and are installed with the help of a release.	Banker et al. (1994); Edberg et al. (2012)
	Change Control	It refers to activities during and after the implementation of new software when the control for the organizational and technological environment is needed.	Baronas and Louis (1988); Duh et al. (2006); Lempinen and Rajala (2014)
Support Control	Performance Control	It refers to monitoring the accomplished tasks of the software (e.g., data flow, and program slicing techniques) for debugging activities.	Banker et al. (1998) Banker et al. (1991); Shaft and Vessey (2006); Weiser (1982)
	Problem Control	It refers to the resolution of emerging problems (e.g., incidents), where clear guidelines of control are needed.	Nelson et al. (2000)

Table 50. Control in IT Operations Projects (Own Depiction)

Best practice frameworks like COBIT and ITIL have provided insights into the basics of service orientation for managers. Although ITIL is quite popular, for example, Addy (2007) mentions that ITIL is very bureaucratic and hardly adapts to fast changing technological environments (Trusson et al., 2014). Furthermore, COBIT concerns the main IS processes. Owing to the fast evolution of new technological and organizational changes, IS control research needs to be expanded (Cram et al., 2016a), for example, to find out how to deal with phenomena like DevOps.

9.2.4 Concepts that Combine Development and Operations

We presented an overview of research on development and operations control. Our findings imply that development control and operations control have different aims. Additionally, these two areas have highly different control philosophies. The collaboration between developers and operations people being investigated for a long time. Markus and Keil (1994) investigated the alignment of system building, business process reengineering, and implementability within IS by highlighting the fact that organizational improvement concerns both system development and IS managers who implement it. Trigg and Bødker (1994) examine how shared PC environments are tailored; they present an understanding of software development tailoring that can be helpful to design systems that can meet customer demands in a better way. Sharma and Rai (2015) analyzed computer-aided software engineering (CASE) as IS innovation, which is defined as a compilation of methods and tools that support an engineering way of development. The major aim of CASE is to improve all the phases of the software development lifecycle (Tate, Verner, & Jeffery, 1992). The CASE approach, however, greatly considers development and explores how the use and integration of tools can support the process (Orlikowski & Daniel, 1991; Tate et al., 1992).

In IS operations control, the literature mainly examines implementation and support control. IT governance frameworks and standards like ITIL and COBIT are used to help organizations to govern their IT tasks. These standards view “what” to achieve good practice through prescriptive guidelines rather than “how”, and the realization often presents challenges to organizations (Boonstra, Yeliz Eseryel, & van Offenbeek, 2017).

Any investigation into issues relating to governance addresses the need for control and collaboration in fast-changing environments but also highlights that it is a contradiction (Sundaramurthy & Lewis, 2003). The relatively underexplored and highly practice-relevant concept of DevOps presents a possibility to resolve the aforementioned challenges and enable better collaboration (Fitzgerald & Stol, 2017). In spite of the attempts to combine software development with operations tasks, we only have a limited understanding of the dimensions of control of the two poles. On the one hand, development often concentrates on a fast way to provide a new software feature within agile software development settings (Cao et al., 2009; Lee & Xia, 2010). On the other hand, operations mainly refers to rigid processes of implementation and support processes (Banker et al., 1998; Boonstra et al., 2017). Research calls for further investigation of the collaboration between developers and operations people (Fielt, Böhmann, Korthaus, Conger, & Gable, 2013; Fitzgerald & Stol, 2017) for enabling teamwork between these poles.

9.3 Research Method

In this section, we present the research design and context, the researcher’s role, the data collection process, and analysis. The influence of IT control in the area of cross-functional DevOps teams is quite unexplored and therefore, we choose an interpretative approach to develop theoretical insights (Ozcan & Eisenhardt, 2009). To investigate our research question, we conducted a multiple case study to identify the control modes of development and operation activities that are integrated within cross-functional teams. We collected qualitative data with the help of a case study and coded as well as analyzed the data according to the grounded theory method (Glaser & Strauss, 1998; Wiesche et al., 2017).

9.3.1 Research Design and Data Collection

Case studies are emphasized for conducting research on "how" or "why" questions by investigating a contemporary issue such as DevOps. Furthermore, case studies are well suited for studying real-life events such as IT governance and control (Yin, 2014). Selecting adequate cases is an essential feature of preparing a theory from case studies (Eisenhardt, 1989). To address our research question on the control dimensions of DevOps teams, we collected primary and secondary data.

We decided to collect primary data for this research and conducted a case study with the help of in-depth multiple case studies to strengthen our knowledge of control in DevOps teams. The unit of analysis is the DevOps team, which develops and operates a service product. We conducted semi-structured interviews with managerial as well as technical employees to identify details relating to the control of development and operations. The investigated firms in our study were different in terms of the numbers of employees, ranging from companies having 501–1,000 employees to more than 30,000 employees. They included firms operating in the retail food, specialized services, non-food, bank, and insurance sector as well as a service company. All the firms had a significant application of the DevOps concept in their IT organization. The informants were carefully selected: the interviewed persons were senior managers, product managers (PO), software developers and engineers, system administrators, and one agile coach. The interviewees were iteratively selected to provide the best information in response to our questions. Questions concerning software management details (i.e., please describe how a new software component runs through several development and operations tasks or please describe how the handover between development and operations tasks is organized) were directed to team members, on the other hand questions on organizational, management, governance and control aspects (i.e., which organizational changes were necessary for integrating the DevOps teams or what are the major changes in leading the team?) were directed to the managers. In addition, we requested the informants to explain their respective IT functions and how the DevOps concept had been integrated. We also asked them to depict their roles and responsibilities as well as relationships with other decentralized functions or units (e.g., the customer). Twenty-one individuals from six firms were interviewed. The interviews typically lasted for about one hour. The interviews were mainly held face-to-face; some interviews were conducted over the telephone. Extensive notes were taken during the interviews (Urquhart, 2012). After every interview, a memo was written down, including follow-up questions for the next interview. Every interview was recorded and transcripts were written. In all, 442 pages of transcripts were generated from the multiple case study and they provide a basis for further analysis.

We also collected secondary data. In particular, we used information on the company that was published on the Internet—for example, on websites, blogs, and annual reports. Several case study participants published information on their company websites and blogs. The main data consisted of the 20 semi-structured qualitative interviews with 21 DevOps team members and managers (one interview was held with two people).

9.3.2 Data Analysis

Qualitative research, especially the grounded theory approach, is chosen to identify the relevant constructs that contribute to the control of development and operations poles in IT teams. Our research started by formulating the problem, and designing and conducting the case study for data collection and analyses (Yin, 2014) for building a grounded theory through the development of abstract categories and the derivation of relationships between them (Urquhart, 2012). The analysis of the transcripts was guided by the Glaserian grounded theory approach of coding (Glaser, 1978). We started the coding process with the help of open coding to derive first insights. We created over 400 codes with respect to control in the 20 interview transcripts. Next, selective coding helped to generate the key dimensions and categories for control in the DevOps teams. Afterwards, we conducted theoretical coding to achieve an understanding of the relationships between these categories by following constant comparison principles (Urquhart et al., 2010). Memos were created from the materials we used to note our new findings and open issues. Those memos were also helpful to iteratively build up our interviews on each other (Urquhart, 2012). At the end, we integrated our theoretical findings with the current theory and domain literature and accordingly delivered a DevOps control model.

9.3.3 Description of the Implementation of DevOps in the Cases

As previously mentioned, our cases are from different industries but all of them have integrated one or more DevOps teams. Table 51 provides a brief overview of how the DevOps concept is integrated in the investigated teams.

Case	Detailed Service Description and DevOps Setting	Interview Roles	DevOps Constellation
FOOD	This involves building and running an app for a delivery service. The team mainly consists of developers; they are responsible for the development of features, conducting continuous delivery, monitoring, and providing support during the daytime. For on-call duty outside normal business hours, decentralized support units are integrated.	Four team members, PO, Agile Coach	Five team members, PO, Agile Coach
BANK	This involves running a securities management system. The provision of tools and a high concentration on automation processes of several environments is integrated. DevOps activities are automated for packaging and the automation of tests at the integration stage, release management, monitoring, end-user communication, and product support (first- and second-level support); third-level support is provided within the developer teams.	Former group manager, group manager, team member, employee	Fifteen team members, Group Manager
INSURANCE	This involves building and running an internal delivery platform for proving infrastructure. Team members have to overtake development tasks like building features as well as operations tasks like maintenance upgrades, capacity scaling, and security fixes. They have rotating responsibilities for	Head of Architecture, team lead, group lead	Eight team members, team lead, Head of Architecture, PO

	on-call duty; they also have to carry out strategic work.		
SERIVCE	This involves building and running an internal delivery platform for proving infrastructure. Team members have to overtake development tasks like building features as well as operations tasks like maintenance upgrades, capacity scaling, and security fixes. They have rotating responsibilities for on-call duty.	Two team leads, Director IT	Eight team members, team lead
SPECIALICED-STORE	This involves building and running an online shop. The members are responsible for the check-out process. The main tasks are frontend and backend development, quality assurance, automation implementation, and operations tasks like monitoring. The team is responsible for on-call duty.	Team lead, team member	Seven team members, PO, team lead
NON-FOOD	This involves building and running an internal platform, which is the substructure of the online shop of the company. On-call duty is organized within the team. Team members are mainly system administrators that overtake the development activities with respect to the platform. The main tasks are developing and providing platform support, integration of continuous delivery, integration of infrastructure as a code, building interfaces for development teams, cooperation with vertical developer teams for testing, and automation of system configuration.	Team lead, two team members	Six team members, team lead/ PO

Table 51. Implementation of the DevOps Concept within the Investigated Teams (Own Depiction)

9.4 Findings

The research process guided us toward the idea of control in DevOps teams. As we depicted earlier in this paper, tensions could appear while combining development and operations tasks in one cross-functional team. Our results present tensions that we derived while integrating cross-functional DevOps in existing IT functions. Furthermore, the process is supported by control for the DevOps setting. Afterwards, we depict insights into how these tensions can be successfully addressed with the help of control modes.

9.4.1 Tensions of DevOps Implementation

Our investigation of control in the DevOps teams was guided by several assumptions. One of our assumptions, which we elaborated in the theory section, is that the application of DevOps is combined with several tensions when the team is not adequately controlled. We develop the idea that IT functions address tensions through control mode portfolios. In Table 52 below, we present six tensions that we derived from empirical data and confirmed them with the help of existing research.

Tension	Description
1. Integrating automation versus manual regulation	On one hand, automation depends on the integration of new tools (Fitzgerald & Stol, 2017; Stein et al., 2013); on the other hand, it depends on the willingness to collaborate with other stakeholders that are used to work with manual steps (Mangalaraj et al., 2009) (e.g., the tools for automation are necessary to implement, and performance lacks can appear if the tool is not fully used because of missing tests and release automation).
2. Initiator versus detector of system failure	On one hand, teams are responsible for operations and support activities for their software (Nelson et al., 2000). On the other hand, problems sometimes cannot be traced back to a particular team or are not recognized by the team itself because the initiator cannot be identified but other stakeholders should spot them and inform the team about the problem for resolving the failure (Lempinen & Rajala, 2014).
3. Claiming self-responsibility versus centralized hierarchies	On one hand, autonomy and decision-making rights have to be handed into teams (McAvoy & Butler, 2009) (e.g., the teams need permission to decide on the technology instead of opening a ticket). On the other hand, centralized hierarchical levels are needed for decisions with strategic impacts and dependencies on the other teams (e.g., budget relevance or technology shifts) (Heumann et al., 2015).
4. Generalist versus specialist knowledge	On one hand, for managing all the tasks of the software delivery lifecycle, a division of tasks and broad knowledge is necessary (Cram et al., 2016a). On the other hand, there are only some persons who have to share knowledge about the tasks of the team and they have to achieve deep domain knowledge (Mähring, 2002; Tiwana, 2009) (e.g. a few team members need to organize many tasks of the team that are not above several highly specialist departments).
5. Combining different backgrounds versus creating a culture of learning	On one hand, developers and operations people have different backgrounds, opinions, and knowledge (e.g., developers want to implement new innovations and software operations people want to have stable systems). On the other hand, an enhancement of their willingness to implement a culture of learning and foster collaboration within the team is needed (Fitzgerald & Stol, 2017).
6. Implementing cross-functional teams versus traditional silo IT	On one hand, cross-functional product-oriented teams enhance the collaboration between stakeholders (Rai & Sambamurthy, 2006). On the other hand, traditional IT functions need to be dissolved and therefore immense efforts and organizational rethinking are needed (Sebastian et al., 2017; Watson-Manheim, Chudoba, & Crowston, 2002).

Table 52. Identified Tensions of DevOps Implementation (Own Depiction)

We argue that these tensions can be resolved through the application of control mode configurations. For developing a shared understanding and terms for the subsequent process analysis of our cases, we derived the dimensions of DevOps control modes by following a grounded theory approach. These dimensions describe what forms of control practices are relevant in DevOps teams with the help of control mechanisms from development and operations perspectives, as shown in Table 53.

Dimensions (D)	Description	Codes for Control Mechanisms
Implementation of Technology Stack (D1)	Implement a technology stack with the help of tools and technology to support development and deployment processes (e.g., configuration management and code reviews) and operations processes (e.g., deployment of code and monitoring)	• Automation processes • Monitoring challenges • Security issues • Tool connectivity • System testing • Configuration/ Deployment management • Infrastructure challenges
Integration of Development and Operations (D2)	Integrate development and operations issues into one team to carry out all the necessary skills, tasks, and processes for planning, building, and running the software delivery lifecycle.	• Development tasks • Use of the agile method • Support issues • Maintenance commitment • Legitimacy concerns • Handover commitment • T-shaped skills expectations
Bridge of Different Cultural Mindsets (D3)	Bridge of different cultural mindsets refers to different backgrounds as well as the willingness of developers and operations people to collaborate, communicate, and trust each other.	• Exchange arrangements • Cultural disconnects • Mind-set convergence • Learning issues • Collaboration issues • On-call duty commitment • Intrinsic motivation • Training mechanisms
Change in Management Style (D4)	Change in the management style refers to the shared roles and responsibilities within the team and the commitment to disciplinarians regarding decision-making processes.	• Autonomy guarantee • Build responsibility enhancements • Handover responsibility • Self-service enabling • Contractual agreements • Leadership consequences
Organization of Product-oriented Teams (D5)	Organization of product-oriented teams refers to the dissolving of classical silo department organizations into cross-functional IT team settings.	• Product orientation • Workplace arrangement • Personal presence agreements • Change management • Top-level support

Table 53. Dimensions of Practices and Mechanisms to Control DevOps Teams (Own Depiction)

The introduction of tensions provides a better understanding of the necessity for control in cross-functional DevOps teams. Based on our findings in our multiple case studies, we developed five resolutions to manage DevOps teams through dynamic control interactions to solve the identified tensions. In the following section, we depict the findings and how the derived tensions can be resolved by implementing DevOps. We show an interplay of the settings of DevOps teams that should be used for managing DevOps teams in existing IT functions. Furthermore, we explain how the different control modes of development and operation, as we depicted in the literature section, can be combined. Finally, we highlight the interactions of control modes and combine them into portfolio configurations for the DevOps teams.

9.4.2 Foster Convergence through Technology to Address Tension 1

Tension automation versus manually regulated installation is resolved and guided through an adequate control configuration. This tension addresses the challenge of combining automated processes as well as control tasks, e.g., manually pushing the button for deployment, which is usually conducted by operations people. Control is automatically integrated in the continuous delivery system. The investigated DevOps teams implemented different continuous processes (Fitzgerald & Stol, 2017) for the automation of software deployments. A DevOps team can push new codes into the production environment with the help of automatic control acceptance steps. The informant explains the advantages of automation: *„We want to reduce the manual activities that people do when they have to operate the platform […]. We show through the results how automation can reduce the time between recovery, how it helps to prevent incidents, and how it helps to improve the quality of the deployment because we do not operate manually any kind of deployment anymore" (INSURANCE, Head of Architecture).*

To support this problem-solving process, significant knowledge of dimensions (see Table 54: Dimensions of DevOps Control), i.e., D1, the implementation of technology tasks such as *automation processes* like test automation, and D2, i.e., a combination of development and operations tasks is necessary. Our results suggest that the development and implementation of automation tools in DevOps teams is based on the combination of behavior control and release control modes. The interaction with development and operations ensures they should work together for integrating "*a packet management system"* (BANK, Team Member) tool, which is necessary for development tasks and testing. Furthermore, the classic release management processes with system outages are eliminated because the *"tasks are appropriate small but we have much better control over what happens within the release"* (SERVICE, Team Lead) since "*you can migrate it with a 'finger snap' and therefore we need no downtime"* (SERVICE, Director IT).

Addressing Tension 1: Integrating Automation versus Manual Regulation

Control Modes	Control Mechanisms
Behavior Control: Controller seeks to influence automation processes to achieve a higher deployment rate. **Release Control:** Controlees enhance the grouping of new software components into packages and installs them in a fully automatic manner.	D1. Implementation of technology stack, e.g.: • Automation processes • Configuration/ Deployment management • System testing D2. Combining development and operations tasks, e.g.: • Handover commitment • Legitimacy concern

Practice Used to Control Tension

- Combination of behavior control and release control:
- Implementation of technology is used to combine automation processes in development (e.g. system package manager) and operations tasks (e.g. deployment management)

Table 54. Resolution 1: Foster Convergence through Technology (Own Depiction)

9.4.3 Identify Dependencies and Differences to Address Tension 2

The tension initiator versus detector of system failure is dispersed through determined team accountability. In classical projects, it is often not clear who is responsible for solving a problem. DevOps solves these tensions as our findings have highlighted that the initiator of the failure is responsible for solving it. Centralized support functions are helpful for DevOps management but the failure needs to be solved by the team.

If problems appear outside the team or are recognized by other stakeholders, contact persons are necessary to manage the problem and identify the responsible DevOps team as mentioned by the following interviewee: *"What is the location of a problem and what does it actually caused? This is the most apparent especially in the interplay between many micro services, which communicate with each other and in principle should be independent of each other. But sometimes there are hidden dependencies concerning data content" (FOOD, Team Member).*

To solve these problems, significant knowledge of dimension D1 change management style is necessary to enable work in DevOps teams, for example, through *autonomy guarantee*. To develop relationships for working in DevOps teams, our findings recommend (D5) *product orientation* as a mechanism.

For integrating these mechanisms, there is a need to combine input control and problem control. The allocation of responsibilities and the associated knowledge to resolve the problems need to be addressed. In the DevOps settings, problem control is integrated in the teams.

Addressing Tension 2: Initiator versus Detector of System Failure

Control Modes	Control Mechanisms
Input Control: The controller defines the working procedures of the controlees. **Problem Control:** The controller has to implement guidelines for problem management.	D1. Implementation of technology stack, e.g.: • Monitoring challenges • Security issues • System testing • Infrastructure challenges D4. Change management style, e.g.: • Autonomy guarantee • Handover responsibility D5. Organization of product-oriented teams, e.g.: • Product orientation

Practice Used to Control Tension

- Combination of input control and problem control:
- Implementation of responsibilities and technologies for managing operations issues (e.g. identification of problems) and delivering the problem to development-related responsibilities (e.g. correct failure in software) is used to guarantee the quick management of failures.

Table 55. Resolution 2: Identify Dependencies and Differences (Own Depiction)

9.4.4 Give Authority to the Guarantor to Address Tension 3

The tension claiming for greater self-responsibility versus centralized hierarchies is addressed by providing authority to the guarantor. In DevOps teams, high responsibility toward the service lies within the team. Hence, DevOps solves this problem through bringing decision-making freedom to the team. Decisions regarding subordinate topics, such as strategic decisions with impact on other teams, need to be separately controlled. The DevOps leadership style is explained by the following quotation: *„If you believe in the DevOps mode and you believe in the tribe [team] concepts, you believe in the autonomy of the tribe. You need to change the leadership style from directive to more collaborative and consensual. For example, the tribe decides who to hire and who to fire. […] we had a situation where the tribe came to me and said, 'This guy is not contributing to our global growth or is slowing down our activities we need to remove him'" (INSURANCE, Head of Architect).*

To control the DevOps team, the dimensions of the change of management style (D4) is necessary. For example, an awareness of *leadership consequences* needs to be known because management gives responsibility to teams. The DevOps team should remain responsible for their software products (D5) and is enabled to make the corresponding decisions. A higher alignment of different cultural backgrounds (D3) is necessary to enhance the willingness to assume service responsibility. Addressing this tension, a combination of clan control and change control is necessary to transfer control to the teams. The integration of flat hierarchies and high self-responsibility is important to make necessary decisions with regard to the service.

Addressing Tension 3: Claiming Self-Responsibility versus Centralized Hierarchies

Control Modes	**Control Mechanisms**
Clan Control: The controller gives responsibility to those teams that share common goals and behave as a group. **Change Control:** Controlees are now responsible for taking decisions on environmental issues, technology, and team enhancements.	D3. Bridge of different cultural mindsets, e.g.: • Cultural disconnects • Mind-set convergence • Collaboration issues D4. Change management style, e.g.: • Autonomy guarantee • Building responsibility enhancements • Leadership consequences • Enabling self-service D5. Organization of product-oriented teams, e.g.: • Product orientation • Change management

Practice Used to Control Tension

- Combination of clan control and change control:
- Transfer self-responsibilities to teams for organizing development and operations activities with the help of flat hierarchies; it is a way to enable decision-making freedom.

Table 56. Resolution 3: Give Authority to Guarantor (Own Depiction)

9.4.5 Broaden Knowledge and Skills to Higher Generalizability to Address Tensions 4 and 5

The tensions generalist versus specialist knowledge and combining different background versus creating a culture of learning are exposed by the resolution to broaden knowledge and skills for higher generalizability. DevOps provides a solution for these tensions because the teams broaden their knowledge of skills to conduct all the necessary tasks of the software delivery lifecycle. Team members expand their knowledge from being a specialist in one area (e.g., either development or operations) to general knowledge, in both areas. Hence, team members develop general knowledge which is necessary for managing the software delivery lifecycle and to overcome culture obstacles by dissolving background constraints. For example, in the NON-FOOD Case, the team is responsible for building and running a platform and the team members mainly have operations backgrounds. Hence, as the interviewee explains, *"it only makes sense that we come together [...] if you are willing you work on the development topic"* (NON-FOOD, Team Lead).

Our findings suggest the use of different control mechanisms (D2), integration of development, and operations tasks to achieve a *maintenance commitment* for the software (D3) as well as bridge between different cultural mind-sets through *collaboration.* A very special procedure of fostering skills in teams was presented by the service case. In our findings, the team lead is responsible for everything that happens in the team and must be able to conduct all the necessary tasks for that service—all these require an immense training effort. This has been illustrated by an informant in the following quotation: *"The team leader is also the system administrator for his application—he/she plays a dual role. In my opinion, this is classic DevOps. [...] The training effort is immersive. [...] Each of these team leaders can represent each other." (SERVICE, Director IT).*

For achieving such a resolution, we recommend combining self-control and performance control for fostering generalized knowledge, which is needed for managing the software. The responsibility has moved from silo-orientation toward team-orientation; team members have to learn new tasks and accordingly need the necessary skills. Less capacity is available for the same spectrum of work.

<table>
<tr><td colspan="2"><b>Addressing Tensions 4 and 5: Generalist versus Specialist Knowledge and Combining Different Backgrounds versus Creating a Culture of Learning</b></td></tr>
<tr><th>Control Modes</th><th>Control Mechanisms</th></tr>
<tr><td><b>Self-Control</b>: Controller fosters the awareness that the team must be able to oversee all the activities of the software delivery lifecycle.
<b>Performance Control:</b> The controlee has to assume responsibility for performance.</td><td>D2. Integration of development and operations, e.g.:
• Development tasks
• Agile method use
• Support issues
• Maintenance commitment
• Legitimacy concerns
• Handover commitment
• T-shaped skill expectations
D3. Bridge for different cultural mindsets, e.g.:
• Cultural disconnects
• Mindset convergence
• On-call duty commitment
• Collaboration issues</td></tr>
<tr><td colspan="2"><b>Practice Used to Control Tension</b>
• Combination of self-control and performance control:
• Implementation of knowledge regarding development and operations parts are used for managing complete service delivery lifecycle activities with only a few people instead of several IT departments.</td></tr>
</table>

Table 57. Resolution 4: Broaden Knowledge and Skills for Higher Generalizability (Own Depiction)

9.4.6 Enhance End-to-End Thinking through Cross-Functionality to Address Tensions 5 and 6

The tensions combining different backgrounds versus creating a culture of learning and implementing cross-functional teams versus traditional silo IT functions can be addressed by enhancing end-to-end thinking in cross-functional teams. IT functions need to integrate cross-functional DevOps teams with organizational and cultural changes. With the help of DevOps teams, a shift from project orientation toward product orientation is enabled. For controlling DevOps our interviews mentioned that product-oriented cross-functional team settings are helpful.

As mentioned by an informant great effort are necessary to integrate cross-functional teams: *„We try to dissolve these vertical silos little by little […]. We would like to work together […] on the product and thus we need to manage our culture more consistently" (FOOD, Agile Coach).* Hence, teams are organized in a product-oriented way of working that enables collaboration and communication for converging different cultural backgrounds. Traditional silo IT functions with high project orientation move toward integrating product-oriented teams. Hence, project management control is not suitable to control DevOps, a control model toward product orientation is necessary. *"Next, we're going to dissolve this organizational silo […]. They, then, go directly into product development teams and are no longer assigned to an organizational unit" (SPEICALIZED STORE, Team Lead).*

Next to the need for integrating the tasks of (D2) development and operations, the implementation of cultural differences (D3) and organization of product-oriented teams with new *workplace arrangements* are consequences. The DevOps teams are organized as product teams that are responsible for one or more services. Hence, a rethink from project management to product management control is necessary. The combination of clan control and change control is a suitable step towards controlling these tensions. Traditionally and historically risen IT functions need to be resolved to enable small and cross-functional teams with product-orientation.

Addressing Tensions 5 and 6: Combining Different Backgrounds versus Creating a Culture of Learning and Implementing Cross-functional Teams versus Traditional Silo IT

Control Modes	Control Mechanisms
Clan Control: Controlees have to share norms and values regarding the service to manage it. **Change Control:** Controllers foster end-to-end responsibilities by changing the environments of organizational setups.	D2. Integration of development and operations, e.g.: • Development tasks • Support issues • Maintenance commitment • Handover commitment • T-shaped skills expectations D3. Bridge for different cultural mindsets, e.g.: • Cultural disconnects • Mind-set convergence • Collaboration issues D5. Organization of product-oriented teams, e.g.: • Change management • Work place arrangement • Product orientation

Practice Used to Control Tension
• Combination of clan control and change control: • The implementation of product-oriented teams responsible for one or multiple service(s) is a way to dissolve organizational silos by considering development and operations.

Table 58. Resolution 5: Enhance End-to-End Thinking through Cross-Functionality (Own Depiction)

9.5 Discussion

We present three key contributions with the help of this research. Firstly, by reviewing the literature and with the help of our case study findings, we were able to identify six main tensions while combining development and operations activities into one team. Secondly, we found that there exist control configurations that address these tensions by implementing adequately combined control modes. Thirdly, for managing these tensions, we derived five resolutions and identified several dimensions of practices and mechanisms to control DevOps teams.

9.5.1 Assessment of the DevOps Control Model

Our main findings present six tensions that can be solved with the help of DevOps. We provide mechanisms and detailed descriptions that address these tensions. We identified five resolutions toward practicing DevOps control that result in different control configurations. The identification of similarities and differences between the resolutions provides a fundamental insight into developing a DevOps control model that explains how controllers can react to tensions over time. Figure 13 presents an overview of the main results and mechanisms; it also shows how these mechanisms are linked to other attributes of the DevOps control model.

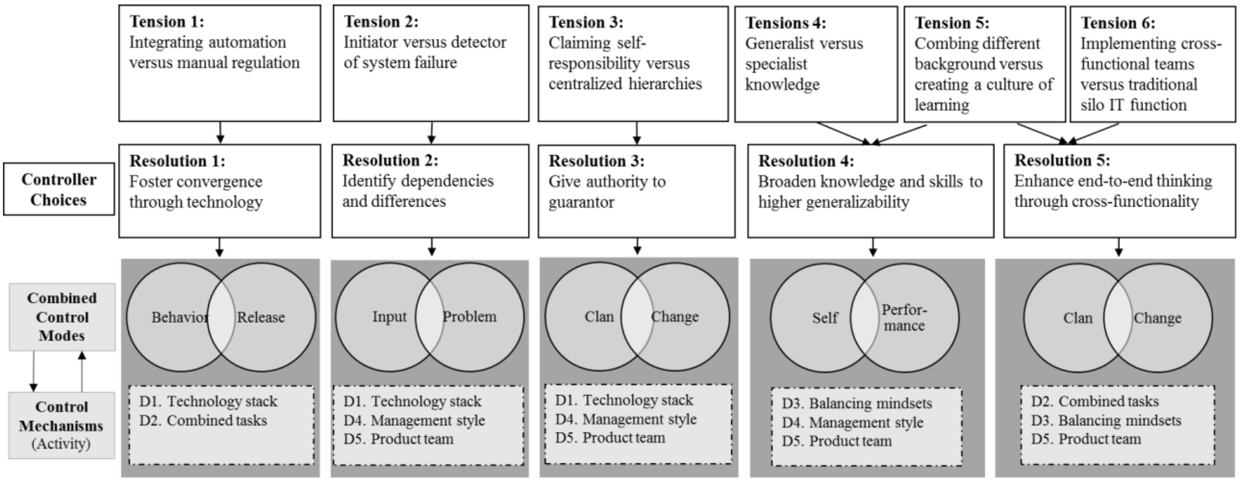

Tension 1: Integrating automation versus manual regulation
Tension 2: Initiator versus detector of system failure
Tension 3: Claiming self-responsibility versus centralized hierarchies
Tensions 4: Generalist versus specialist knowledge
Tension 5: Combing different background versus creating a culture of learning
Tension 6: Implementing cross-functional teams versus traditional silo IT function
Controller Choices
Resolution 1: Foster convergence through technology
Resolution 2: Identify dependencies and differences
Resolution 3: Give authority to guarantor
Resolution 4: Broaden knowledge and skills to higher generalizability
Resolution 5: Enhance end-to-end thinking through cross-functionality
Combined Control Modes
Behavior
Release
Input
Problem
Clan
Change
Self
Perfor-mance
Clan
Change
Control Mechanisms (Activity)
D1. Technology stack
D2. Combined tasks
D1. Technology stack
D4. Management style
D5. Product team
D1. Technology stack
D4. Management style
D5. Product team
D3. Balancing mindsets
D4. Management style
D5. Product team
D2. Combined tasks
D3. Balancing mindsets
D5. Product team

Figure 13. Summary of Findings: A DevOps Control Model (Own Depiction)

Each presented resolution depicts a proposal on how a tension can be resolved. The solving processes for these tensions can be seen as independent from each other. The derived DevOps control model is discussed in more detail below:

9.5.2 Differences in Control of DevOps

Our results suggest that the control of DevOps relies on control portfolio configurations. We conceptualize DevOps control by making frequent adjustments along the interplay between different control attributes: combining control modes and control mechanisms to address the identified tensions. Furthermore, we derived resolutions on how a controller can react to tensions. This differs significantly from prior research, which primarily pays attention to the contextual antecedents and performance consequences of control modes as well as on the degree of control portfolio configuration (Choudhury & Sabherwal, 2003; Kirsch, 1996; Kirsch, 1997). Thus, the grounded theory of DevOps control in internal IT teams—introduced in this paper—explains the coexistence of the controls that are used on the basis transferred to practices. Our DevOps control model, presented in Figure 13, provides insights into the control for DevOps teams; this can be seen as a dynamic and iterative DevOps control model and not as a linear process for addressing tensions and the creation of control portfolios over time (Choudhury & Sabherwal, 2003; Gregory et al., 2013; Kirsch, 2004).

Control modes are an attribute of controlling the DevOps teams, which emerged from our data. We found that for the management of DevOps teams, a combined approach of control modes of IS development project control and operations control are necessary. Our findings highlighted that both control modes need a tighter connection for controlling DevOps teams. Combining development and operations activities in a single cross-functional team requires an approximation of both control perspectives. Our findings present four modes of combining control that appear during the DevOps control model: behavior-release, input-problem, clan-change, and self-performance. No combination with respect to the outcome control and operational control was found. In outcome control, the controller concentrates on the interim and final outcome that is delivered by the project team. Project objectives are defined by the controller and the team is rewarded if these objectives are achieved (Kirsch, 1997). Our results depict that DevOps teams work very autonomously with high self-responsibility and organize their objectives and how they are achieved by themselves. This is one reason why outcome control does not appear in our DevOps control model.

Control mechanisms are another attribute of our DevOps control model. Our results suggest that the control of DevOps relies on a portfolio of formal and informal project control (Table 49) as well as implementation and support operations control (Table 50). A row of control mechanisms was introduced in this paper, but some other dominant mechanisms include *automation processes, mindset convergence, autonomy guarantee,* and *product orientation.*

Automation processes are used as the basis for implementing DevOps technologies for managing tasks between developers and operations people. Through a high-level of automation, a major advantage of speed and agility during the development and release processes can be achieved with the help of test automation and manual acceptance test, which are often rather time-consuming and can be avoided. It is important to familiarize the automation processes with a change or convergence of the *mindset* of the stakeholder. People working within a DevOps team or collaborating with them must be open-minded to this new form of work. The controller claims a higher understanding of how the DevOps approach works and its advantages for the organization.

Product orientation is implemented for resolving the conflicts that often appear due to insufficient communication between the development and operations departments. Enabling work within the team setting rather than in separated departments leads to better collaboration and communication between team members. Conflicts can be resolved by product-oriented cross-functional teams. We observed that a high level of *autonomy* is *guaranteed* within the DevOps teams. Controllers must decide how much autonomy must be given to the teams. For enhancing decision-making regarding tools, self-services for DevOps are implemented. Hence, the team can decide when they need new technology, such as a database, instead of writing emails and open tickets to support functions (Nelson et al., 2000).

9.5.3 Transformation of the Control Idea from Project to Product Management

A major finding of our study regarding control in cross-functional teams is that DevOps teams do not usually work in a project setup: they use so-called product setups. IS projects usually have pre-defined project goals, such as on-time and within-budget delivery (Kirsch, 2004). Our findings highlight that there is a shift toward product-orientation—this means that there is no starting and ending date for the product team. Furthermore, budget and on-time milestones are not controlled by the team in our investigation. A redesign of the processes and end-to-end IT service responsibility is needed within cross-functional teams (Balaji et al., 2011). However, as already outlined in this paper, there is a need for control in cross-functional teams. Research on DevOps should consider that other success and failure measurements as well as control structures and mechanisms are necessary in this regard.

9.6 Conclusion

This study elucidates the dynamic and iterative nature of the control of internal DevOps teams. DevOps is a phenomenon that is quite unexplored (Sebastian et al., 2017). Implementing DevOps is associated with control adaptation and how controllers interact with the DevOps team. We started with the identification of tensions that appear in cases between development and operations while DevOps is not adequately managed. Six tensions were derived with the help of existing literature and empirically confirmed through our research. Furthermore, we presented an investigation of control configuration by focusing on combining control modes and control mechanisms to react to these tensions. For managing tensions, five resolutions for controlling DevOps were introduced to gain a richer understanding of how control activities differ for addressing tensions in the team. In this paper we have developed dimensions of practice summarized in the resolutions as a core for control in DevOps teams following the grounded theory approach. These dimensions describe the forms of control practices that are relevant in DevOps teams through control mechanisms with regard to development and operations perspectives. In addition, results present evidence that a combined approach of control modes of development and operations is necessary. Furthermore, we explain the interaction of control portfolio configurations and hence, depict how control is arranged in DevOps teams. This research resulted in a DevOps control model. The DevOps control model explains an iterative approach to controlling cross-functional teams for managing challenges.

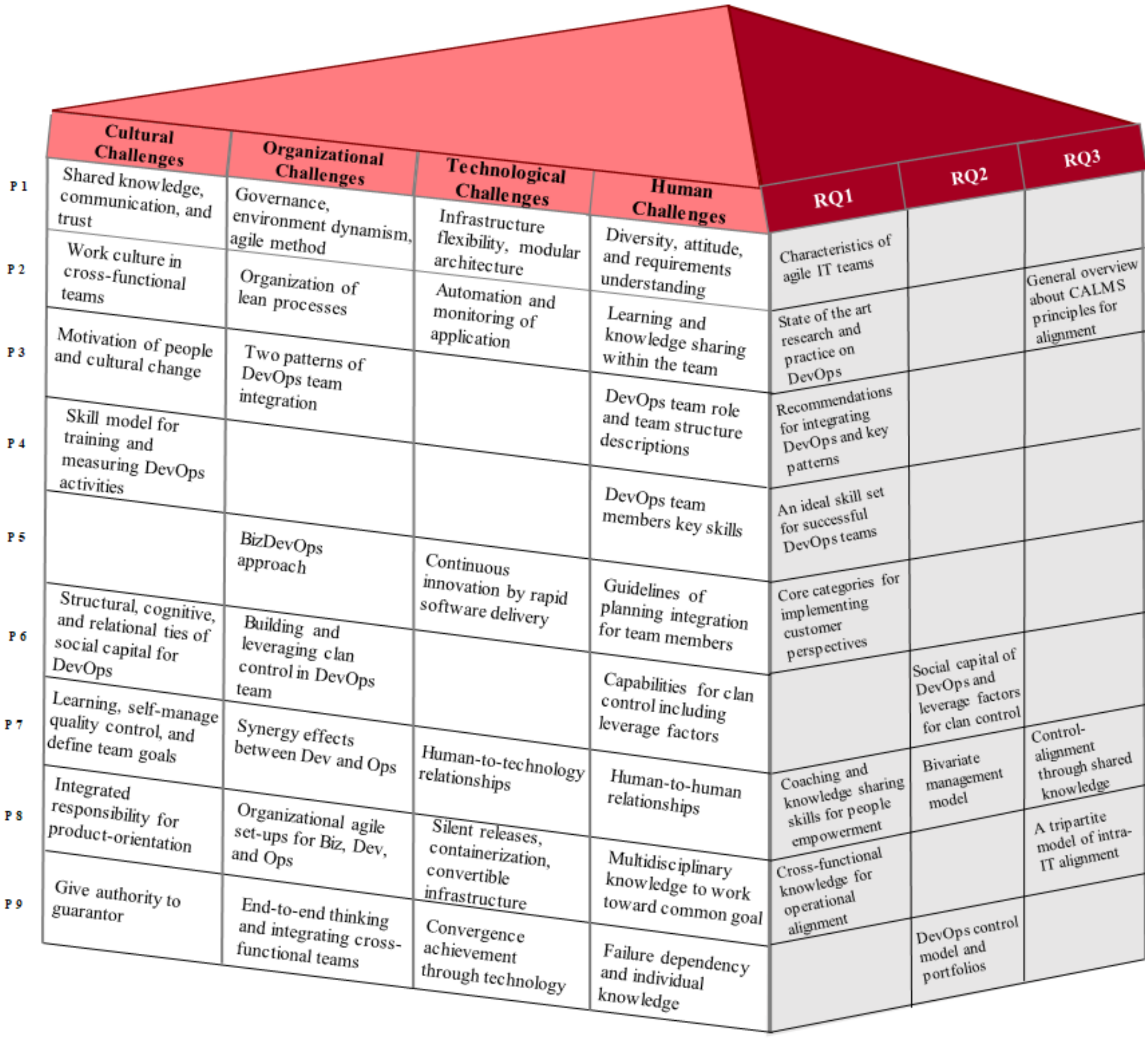
In summary, we enhanced the existing literature through the presentation of four new modes of control for DevOps teams, namely behavior - release, input - problem, clan - change, and self - performance. Furthermore, we provided insights into the neglected area of control in cross-functional teams and provided initial insights into the interactions between controller and controlee to integrate control configuration in practice. Thus, our grounded theory of DevOps teams' control in internal IT functions shows new ways of the control phenomenon in intrafirm relationships.

PART C

1. Main Research Findings

The overall aim of this dissertation is to present theoretical foundations and empirical insights into how organizations can react to, manage, and govern the challenges of cross-functional teams.

This cumulative dissertation includes nine papers. These papers vary by theoretical background, research method, findings, and contributions. Nevertheless, all nine papers aim to address one or more of the challenges introduced in chapter 1.2 as well as one or more research questions presented in chapter 1.3. The findings and contributions of this thesis are presented in Figure 14. The figure presents a holistic overview of a unique set of examinations (acronym: HOUSE). This DevOps HOUSE serves as an initial overview of the main research findings followed by more detailed explanations in the next sections.



	Cultural Challenges	Organizational Challenges	Technological Challenges	Human Challenges	RQ1	RQ2	RQ3
P 1	Shared knowledge, communication, and trust	Governance, environment dynamism, agile method	Infrastructure flexibility, modular architecture	Diversity, attitude, and requirements understanding	Characteristics of agile IT teams		
P 2	Work culture in cross-functional teams	Organization of lean processes	Automation and monitoring of application	Learning and knowledge sharing within the team	State of the art research and practice on DevOps		General overview about CALMS principles for alignment
P 3	Motivation of people and cultural change	Two patterns of DevOps team integration		DevOps team role and team structure descriptions	Recommendations for integrating DevOps and key patterns		
P 4	Skill model for training and measuring DevOps activities			DevOps team members key skills	An ideal skill set for successful DevOps teams		
P 5		BizDevOps approach	Continuous innovation by rapid software delivery	Guidelines of planning integration for team members	Core categories for implementing customer perspectives		
P 6	Structural, cognitive, and relational ties of social capital for DevOps	Building and leveraging clan control in DevOps team		Capabilities for clan control including leverage factors		Social capital of DevOps and leverage factors for clan control	
P 7	Learning, self-manage quality control, and define team goals	Synergy effects between Dev and Ops	Human-to-technology relationships	Human-to-human relationships	Coaching and knowledge sharing skills for people empowerment	Bivariate management model	Control-alignment through shared knowledge
P 8	Integrated responsibility for product-orientation	Organizational agile set-ups for Biz, Dev, and Ops	Silent releases, containerization, convertible infrastructure	Multidisciplinary knowledge to work toward common goal	Cross-functional knowledge for operational alignment		A tripartite model of intra-IT alignment
P 9	Give authority to guarantor	End-to-end thinking and integrating cross-functional teams	Convergence achievement through technology	Failure dependency and individual knowledge		DevOps control model and portfolios	

Figure 14. The DevOps HOUSE (Own Depiction)

The main research results of each paper are summarized blow.

Paper 1: Intellectual Capital Within Cross-Functional IT Teams

There has been extensive research into agile development teams, but less is understood about how to enable IT teams to develop organizational learning. Prior research on organizational learning underscores its importance for realizing change (Kang & Snell, 2009). In response to external changes (i.e., the capability to be agile), IT teams need to perform ambidextrous learning: exploring new knowledge while exploiting existing knowledge (Lee et al., 2015). Based on the dimensions of intellectual capital (organizational, human, and social capital), Paper 1 depicts intellectual capital configurations and contributes to theory by introducing the new dimension of technological capital. The findings present insights from a systematic literature review and highlight eleven key characteristics with respective descriptions of cross-functional teams. The results of Paper 1 present capabilities that enable cross-functional agile IT teams to manage key challenges, as illustrated in Figure 15.

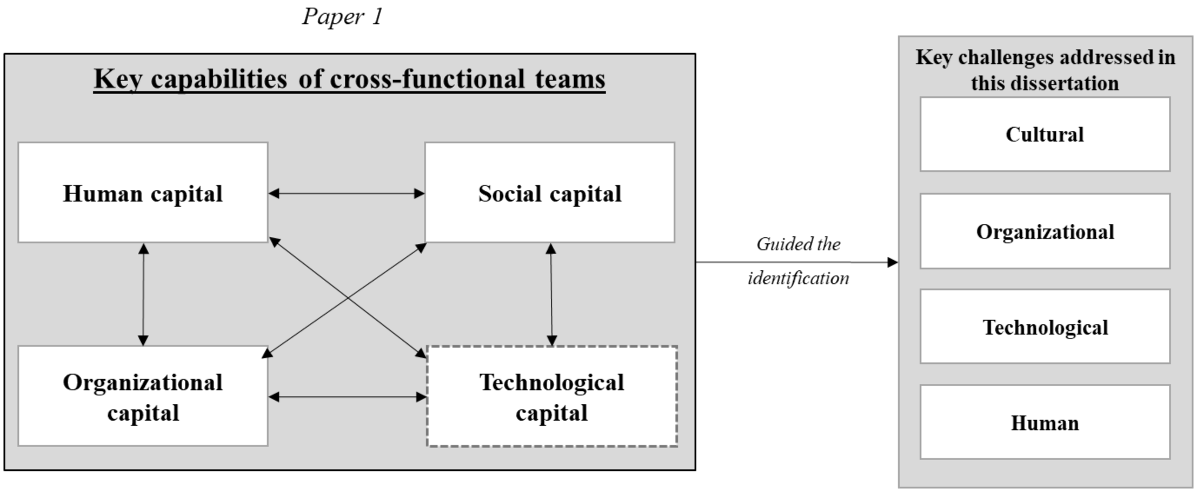


Figure 15. Findings of Paper 1 (Own Depiction)

Paper 1 represents the theoretical foundation of this dissertation because, the key challenges were developed through this investigation. The results of this study show the main characteristics based on human, social, organizational, and technological capital, and they lead to deriving the corresponding challenges. The results of this thesis suggest that addressing the cultural (social challenges were extended and renamed to cultural), organizational, technological, and human challenges, which are explained in Part A, chapter 1, will lead to efficient and well-aligned DevOps teams.

Paper 2: The DevOps Phenomenon: An Executive Crash Course

Since the first DevOpsDays conference took place in Ghent, Belgium in October 2009 (Reed, 2014), the DevOps phenomenon has steadily gained popularity (Debois, 2011). Research and practice continue to collect insights and integrate this concept into established organizations (Hemon et al., 2018). Several researchers have called for further investigations of DevOps (Hemon et al., 2018; Krancher et al., 2018) and several attempts have been made to define

DevOps (e.g. Forsgren (2018) or Roche (2013). Paper 2 consolidates and presents an overview of extant DevOps research and develops a research for practice approach that includes comprehensive guidelines, practical challenges, opportunities, and case study examples for managers as well as academics.

Paper 2 addresses RQ1 by defining DevOps principles, required capabilities, and guidelines for managing the challenges incumbent firms face. Paper also addresses RQ3 by explaining the CALMS principles in detail and by showing how integrating CALMS principles to a high degree facilitates DevOps team alignment. Paper 2 also investigates the role of work culture and makes recommendations for communication and building relationships. It finds that lean processes reduce organizational challenges by minimizing work in progress and that identifying the tools necessary to support the software delivery lifecycle reduces technology challenges. Finally, it identifies personal learning and willingness to share knowledge as human challenges.

Paper 3: Integrating DevOps within IT Organizations– Key Pattern of a Case Study

The digital transformation impacts the organizational setup of IT functions (Sebastian et al., 2017). For the fast delivery of new software features and to react to changing customer requirements, IT functions have started integrating cross-functional teams (Ross et al., 2016). Little is known about how IT functions arrange cross-functional teams within their traditional organization. Big established organizations are typically used to working in silo-oriented functions (Forsgren et al., 2017) with strong governance structures and now they are challenged to integrate autonomous self-organized teams (Gregory et al., 2018). Paper 3 contributes to research by identifying key patterns of DevOps setups and possible DevOps team organizations for incumbent companies. Paper 3 distinguishes two pattern of DevOps teams: platform-oriented, and application-oriented teams. Furthermore, the paper provides insights into DevOps tasks and four starting points for incumbent IT functions to integrate DevOps in different situations and contexts.

Paper 3 addresses RQ 1 by providing key patterns and insights of possible DevOps setups and starting points. Hence, the paper analyzes DevOps teams and how incumbent firms can motivate people and enable cultural change. Furthermore, the two key patterns present guidelines for reorganizing IT functions and provides insights into cultural challenges. Finally, its insights into DevOps tasks and team roles help address human challenges.

Paper 4: Are You Ready for DevOps? Required Skill Set for DevOps Teams

Incumbent organizations are beginning to implement cross-functional agile teams to address current challenges and empower future-proof service delivery (Jöhnk et al., 2017). Agile software development methods addresses mainly the relationships between software development and business units (Lwakatare et al., 2016; Tripp et al., 2016). Initial investigation into DevOps presents first insights of capabilities (Lwakatare et al., 2016), but a comprehensive overview on required skills is lacking. To fill this research gap, a workshop with IT consultants and additional expert interviews with consultants were conducted. The findings of Paper 4 constitute a comprehensive overview of seven skill categories with detail description and 36 concrete key skills of DevOps teams. The seven skill categories are: full-stack development, analysis, functional, decision-making, social, testing, and advisory.

Paper 4 contributes to answer RQ 1 by providing an overview on key skills for DevOps team members. This paper finds that many skills relate to human challenges and provides recommendations for integrating an ideal skill set. This research presents a model for training and measuring DevOps people, which leads to a cultural movement of mindsets.

Paper 5: Implementing the Planning Process within DevOps Teams to Achieve Continuous Innovation

Cross-functional teams integrate the business perspective and broaden agility to business functions (Holmqvist & Pessi, 2006). IT managers are challenged to develop business capabilities and business managers must develop IT capabilities (Recker et al., 2009). DevOps and BizDevOps approaches are contemporary concepts that can solve these IT and business capability development needs (Fitzgerald & Stol, 2017). A key aspect of the BizDevOps mode is continuous planning, which enables IT and business to work together (Fitzgerald & Stol, 2014; Silverthorne, 2017). Less is known about how to integrate the planning process and the customer view in cross-functional teams. The findings of Paper 5 are based on an explorative case study with 23 expert interviews. The research identified scalability, security, and quality of planning as mechanisms for continuous innovation in BizDevOps teams.

Paper 5 addresses RQ 1 by identifying core categories for planning and by underscoring the need to integrate business capabilities into cross-functional IT teams. Incorporating BizDevOps concepts and a high level of business integration will help solve organizational challenges. The paper finds that continuous innovation affects technological challenges through rapid software delivery and guidelines for integrating planning processes affect human challenges.

Paper 6: How to Implement Clan Control in DevOps Teams

Control is a method of aligning individual human behavior with organizational aims (Wiener et al., 2016). Creating control capabilities in internal IT teams is required to achieve common communication and coordination among autonomous self-managing teams (Gregory et al., 2018). Informal clan control influences team members by creating shared values and norms, and accelerates feelings of affiliation by setting common group goals (Kirsch et al., 2010). Extant literature finds that social capital positively influences teamwork and performance (Liu et al., 2015) but more research is needed into overcoming barriers related to the different backgrounds of software developers and operations people. Paper 6 addresses this gap and contributes to existing literature by examining team-based control through the application of social capital resulting in leverage factors for DevOps teams. The findings can serve as guidelines for managers to enhance clan control within teams.

Paper 6 addresses RQ2 by presenting social capital of DevOps characteristics and leveraging factors for clan control. These findings serve as control mechanisms for guiding DevOps teams. Paper 6 addresses cultural challenges by identifying the importance of social capital ties for DevOps teams. Additionally, building and leveraging clan control in DevOps fosters organizational transformation of DevOps teams. Finally, recommendations for solving human challenges are suggested by capabilities for clan control as part of social capital

Paper 7: A Control-Alignment Model for Product Orientation in DevOps Teams – A Multinational Case Study

IT functions require new forms of governance arrangements to manage digital services and product networks (Ross et al., 2016). Integrating flexibility and control in software development and delivery is a key concern of incumbent IT functions (Lamb & Kling, 2003). Organizations seek to encourage bidirectional governance and knowledge sharing (Tiwana et al., 2013) through combining software developers' and software administrators' perspectives into one cross-functional team (Krancher et al., 2018). Prior examinations of control-alignment concentrate on different software development project setups (e.g., Cram et al. (2016b)) or digital infrastructure governance and control (e.g., Yoo et al. (2010)). Despite recent developments in product-oriented software, little research has been conducted into how changing control and governance structures affect product orientation in cross-functional IT teams.

The results of Paper 7 are based on a multinational multiple case study and additional expert interviews with seven thought leaders and industry experts. 42 interviews were conducted. The results of this multinational multiple case study depict a new, bivariate management model that has its roots in control and alignment theory. Paper 7 identifies shared domain knowledge as a critical enabler of control in DevOps teams, and it sheds light on the human-to-human (controlled-by-competence) and human-to-technology (controlled-by-invisibility) control-alignment mechanisms of DevOps teams. The bivariate management model is illustrated in Table 59 with the help of a "Control-Alignment Tetrahedron". This paper finds that shared domain knowledge catalyzes control mechanisms resulting in bivariate management. The DevOps tetrahedron can present different stages of alignment ranges from highly aligned control-alignment in DevOps to highly misaligned control-alignment. Paper 7 focuses on the highly aligned version of the control-alignment DevOps tetrahedron. Table 59 provides examples and descriptions of the characteristics of control alignment.

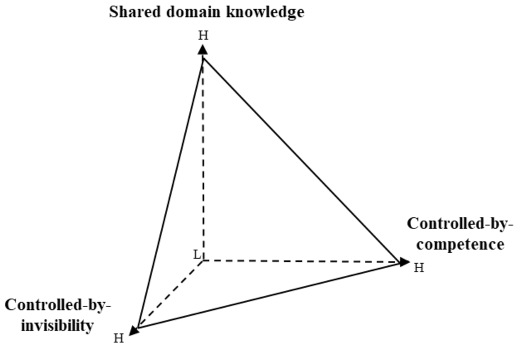
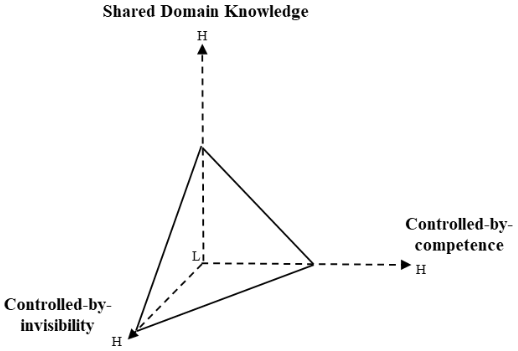
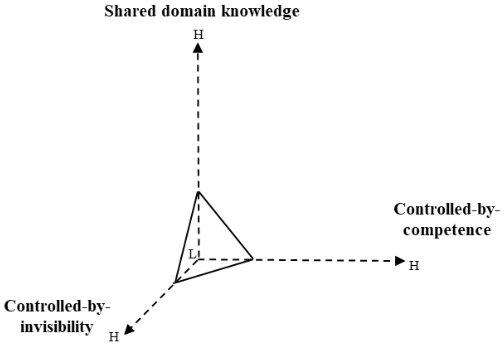
Control-Alignment Tetrahedron	Description
 Shared domain knowledge H Controlled-by-competence H L Controlled-by-invisibility H	**Highly aligned DevOps Tetrahedron** The tetrahedron depicted on the left illustrates the optimal condition of bivariate management. Control mechanisms of both control categories are integrated and enabled by a high level of shared domain knowledge in this DevOps team.
 Shared Domain Knowledge H Controlled-by-competence H L Controlled-by-invisibility H	**Partial (mis-)aligned DevOps Tetrahedron** The tetrahedron depicted on the left illustrates a partial misaligned form of bivariate management. Human-to-technology mechanisms (controlled-by-invisibility) are highly integrated in this example but human-to-human mechanisms (controlled-by-competence) are medium distinct. A medium level of shared domain knowledge is practiced in this DevOps team.
 Shared domain knowledge H Controlled-by-competence H L Controlled-by-invisibility H	**Highly misaligned DevOps Tetrahedron** The tetrahedron depicted on the left constitutes the suboptimal condition of an aligned and controlled bivariate management. Control mechanisms of both control categories are not high and the level of shared domain knowledge is low in this DevOps team.

Table 59. Example Constellations for (Mis-)Alignment (Own Depiction)

Paper 7 addresses RQ1 by providing insights for DevOps team members and executives regarding coaching and knowledge sharing skills for people empowerment. The research contributes to RQ2 by showing how the bivariate management model provides dedicated control mechanisms and by identifying the two control core categories controlled-by-invisibility and controlled-by-competency of DevOps teams. In addition, Paper 7 addresses RQ3 by showing how shared knowledge facilitates long-term alignment. In terms of cultural challenges, this investigation introduces learning and self-manage quality control and defines team goals. Organizational challenges are addressed by the achievement of synergy effects between development and operations in cross-functional teams. The two core categories of bivariate management offer recommendations for solving technology and human challenges.

Paper 8: Understanding how DevOps Aligns Development and Operations: A Tripartite Model of Intra-IT Alignment

IT-business alignment is still a core concern of IT functions (Gerow et al., 2014a). Closer collaboration between the IT subunits' development and operations are key to achieving agility and alignment during software development and delivery (Krancher et al., 2018). Paper 8 concentrates on the need to align internal IT development and operation subunits, which traditionally pursue different goals and are often misaligned (Onita & Dhaliwal, 2011). Traditionally, software development and software operations are separated in different IT units, but IT executives become aware of the advantages of DevOps and start to integrate such teams within their organization (Hemon et al., 2018; Krancher et al., 2018). Prior research identifies the need to include software operations within an agile environment (Krancher & Luther, 2015). The findings of Paper 8 are based on a multiple case study with 8 participating cases including 26 expert interviews. Paper 8 identifies three mechanisms that serve as a precondition for developing intra-IT alignment.

Paper 8 addresses RQ1 by providing an overview of cross-functional knowledge for operational alignment, and it addresses RQ3 by providing a model for operational alignment within the IT function. Furthermore, Paper 8 presents recommendations for solving all described challenges. The developed model provides insights into how alignment between development and operations can be realized in DevOps teams.

Paper 9: Combining Development and Operations: Toward a Control Model in Cross-Functional Teams

Research in IS has predominantly investigated control in project setups (Eseryel & Eseryel, 2013; Wiener et al., 2016). Some examinations focus on either control in software development (Maruping et al., 2009) or in software operations (Banker et al., 2002; Nelson et al., 2000), but leave it to further research to combining the two control activities in cross-functional teams. To date, little research is available into understanding and resolving tensions between software developers and software operations (Edberg et al., 2012). To fill this research gap, a multiple case study with 6 companies including 21 interviews with DevOps experts was conducted. The results of Paper 9 contribute to research by building on the control modes of development and operations. They depict combined control modes for solving tensions and achieving suitable control configurations for managing DevOps teams. The findings present a new model for the design and implementation of control modes.

Paper 9 contributes to RQ2 by developing a new DevOps control model with control portfolios rooted in control theory supported by empirical evidence from the multiple case study. All six challenges are addressed by the recommendations of giving authority to guarantor (cultural), fostering end-to-end thinking and integrating cross-functional teams (organizational), achieving convergence through technology within DevOps teams (technological), and integrating individual knowledge (human).

Summary of Finding

Research results of the publications included in this dissertation are presented in Table 60.

Publication	Main Results
P 1	• Characteristics of cross-functional IT teams enable ambidextrous organizational learning with the help of intellectual capital. o Evidence for simultaneously conducting exploration and exploitation in the context of intellectual capital o Configuration that supports organizational learning of cross-functional IT teams. o Manifestations for social, organizational, human, and technology capital
P 2	• Practice-oriented preparation of the DevOps concept (research for practice) o Promises and challenges of DevOps for digital transformation o Explanations of the movement toward product management o Overview on DevOps principles and state-of-the-art definition and similar extant research o Practical challenges, opportunities, and case study examples
P 3	• Key pattern of DevOps setup (platform and application teams) • Starting points for DevOps integration • Explanations for product orientation
P 4	• Concept of skills for DevOps teams o Investigation of attribute-based competencies o Seven skill categories (with full-stack development, analysis, functional, decision-making, social, testing, and advisory skills) with 36 concrete skills o Skills model for human resource development and performance measurement
P 5	• Integrating the business perspective with the help of planning processes within DevOps teams o Introducing the BizDevOps Concept o Integrating business view with the help of a product owner in the team, on the business side or as a team lead. o Presenting areas of planning (planning responsibility, planning scope, and planning dependency) o Deriving mechanisms for continuous innovation and planning relation (scalability, security, and quality)
P 6	• Explanations for integrating clan control in DevOps teams o Deriving of structural, cognitive, and relations ties of social capital within DevOps teams o Leverage factors for facilitating clan control o Showing recommendations for integrating these factors
P 7	• Control-alignment model in DevOps teams o Deriving of the bivariate management model o Introduction of controlled-by-competencies and controlled-by-invisibility as core control categories

	o Presenting shared-domain knowledge as a catalyst for control-alignment o Illustrating bivariate management as control-alignment tetrahedron
P 8	• Intra-IT alignment within DevOps teams • A Tripartite Model of Intra-IT Alignment o Three mechanisms to achieve alignment within DevOps teams: individual componentization, integrated responsibility, and multidisciplinary knowledge o Model of intra-IT alignment explains resolving misalignment o Integration into operational alignment research
P 9	• DevOps control model to address tensions while combining software development and operations. o Presentation of six types of tensions while combining development and operations tasks in one cross-functional team o Introducing evidence for combining control modes from software development and operations o Highlighting recommendations for solving these tensions

Table 60. Summary of Findings of Publications Included in this Thesis (Own Depiction)

2. Discussion

2.1 Contributions to Theory

This dissertation has been guided by three research questions as outlined in Part A, chapter 1. This section depicts how the main research findings adds value to the existing theories of skills, control, and alignment.

2.1.1 IS Capabilities and the Concept of Skills

This dissertation contributes to concepts of skills and competencies in numerous ways. Digital transformation leads to the development of new skills and capabilities within the IT functions and teams (Jöhnk et al., 2017). The implementation of the DevOps concept requires effort by building and creating new values by linking existing capabilities with new ones (Ross et al., 2016). For incumbent firms, it is not clear what skills are required to manage the different tasks expected from a cross-functional DevOps team. To fill this gap, this research defines an ideal skillset for successful teams. Therefore, this investigation provides an understanding of the attribute-based competencies of DevOps team members, sheds light on DevOps team profiles and makes recommendations for integrating DevOps skills. By providing core skill categories and related skills, this examination enhances existing literature by applying concepts of skills to a new phenomenon. Additionally, guidelines for designing trainings to develop highly skilled IT people are valuable for business (human resource) and IT.

Existing literature identifies knowledge gaps between business and IT people (Markus & Keil, 1994). BizDevOps can achieve convergence between these different knowledge levels. Scholar outline the need for further research of DevOps teams to achieve continuous planning and innovation capabilities (Fitzgerald & Stol, 2017; Sebastian et al., 2017). In response to these calls, three areas of planning are depicted in this dissertation that enhance continuous innovation through BizDevOps. In addition, three dimensions of planning possibilities are provided for explaining the integration of the business view into cross-functional teams.

Furthermore, this investigation builds on prior research by highlighting the necessity of knowledge sharing skills and cross-functional knowledge within DevOps teams. To align people with software development and software operations backgrounds (Edberg et al., 2012), the willingness to share knowledge and mutual learning is essential. This research outlines that DevOps teams need strong cross-functional knowledge. Operational alignment literature emphasizes that social capital enables knowledge sharing (Wagner et al., 2014). DevOps teams are responsible for problem management and are motivated to solve problems proactively and develop a high skill level in satisfying customer demands.

2.1.2 Control Theory

This dissertation contributes to control theory in several ways. Papers 6-7, and 9 provide numerous novel findings in the areas of leveraging clan control in DevOps teams, combining software development and operations control modes, and control mechanisms for DevOps teams resulting in a bivariate management model. This dissertation builds upon both control theory perspective, software development and software operations. Extant research argues often convincingly for separating these control perspectives (e.g., Kirsch et al., 2002; Nelson et al., 2000). Nevertheless, external influences and changing customer demands require adaption in IT governance and control as organizations move from functional silo units toward self-managed cross-functional teams (Gregory et al., 2018). Control modes like change control in operations were necessary to control insufficient developments before deployment (Duh et al., 2006; Lempinen & Rajala, 2014). In DevOps teams, frequent deployments with small amounts of new code are deployed that are less complicated. Hence, adjustments of control are needed to effectively manage DevOps teams. This dissertation contributes to this discussion in three ways:

First, this thesis gives novel insights into managing clan control in cross-functional DevOps teams. Prior literature depicts that social capital enhances collaboration and teamwork (Liu et al., 2015). To facilitate strong teamwork, more and more managers are bringing team members closer together and implementing team structures (Chua et al., 2012; Majchrzak et al., 2005). Overcoming differences between DevOps team members' backgrounds is key to efficient collaboration (Fitzgerald & Stol, 2017). Scholars have found that working in complex environments requires strong collaboration in cross-functional teams, which can be facilitated by developing shared understanding, shared values, and social cohesion (Liu et al., 2015; Majchrzak, More, & Faraj, 2012). Furthermore, clan control enabled by social capital is required at the team level to foster collaboration. This dissertation points to potential benefits of clan control in DevOps teams and describes how managers can foster social capital to achieve clan control in DevOps teams. Building upon the work of Liu et al. (2015) and Chua et al. (2012), this dissertation confirms that a manager can be a part of the clan. In addition, this research derives three leverage factors to accelerate clan control in DevOps teams: call for operations responsibility, integrate a culture of feedback and learning, as well as establish receptiveness for fast changes. This research contributes to the prior findings of Liu et al. (2015) and Chua et al. (2012) and follow their call to apply their concepts in different project setups. Research in the area of clan control in cross-functional teams is enhanced by including DevOps-specific social capital and the three leverage factors for managing and achieving clan control.

Second, this research contributes to control theory by depicting control mechanisms that guide software development and operations activities in cross-functional teams. This dissertation proposes control modes for DevOps team's management: behavior-release, input-problem, clan-change, and self-performance modes. Extant investigations have a strong focus on software development project control (Maruping et al., 2009) but only little research provides insights into combining software development and software operations control (Edberg et al., 2012; Goh et al., 2013), which is necessary for cross-functional teams. This dissertation fills this research gap by offering findings about combining different control modes of software development and software operations to solve and control challenges resulting from integrating DevOps teams. This research outlines how five resolutions comprising different control configurations contribute to developing a DevOps control model and outlines the benefit of control portfolio configurations, including control modes and control mechanisms from software development and operations in resolving challenges in DevOps teams. Going beyond extant literature, which predominately focuses on contextual antecedents and performance consequences of control modes and the extent to control portfolio configuration (Choudhury & Sabherwal, 2003; Kirsch, 1997), this research shows how controllers can react to DevOps-related tensions. The DevOps control model is a dynamic and iterative model for controlling tensions and creating control portfolios with a long-term perspective. This study enhances existing inquiries through the presentation of new control modes for DevOps teams grounded in software development and software operations control theory, namely: behavior - release, input - problem, clan - change, and self-performance. Furthermore, this research casts light on the neglected research area of cross-functional teams in intrafirm relationships.

Third, managing DevOps teams require a transformation of separated development and operations control to joint control of DevOps teams to enable high performance in cross-functional working modes (Hemon et al., 2018). By combining software developers' and software operations' perspectives into one self-reliant team (Krancher et al., 2018), firms seek to achieve bidirectional control and knowledge sharing (Tiwana et al., 2013). Prior literature identify knowledge as a required element of organizational control (Ouchi, 1979; Tiwana & Keil, 2007). Notably, for internal IT functions to manage effectively the software delivery lifecycle as a self-reliant team, alignment of IT knowledge across multiple subunits (e.g., software development and software operations) is needed. This dissertation investigates alignment in DevOps teams, and offers recommendations on how cross-functional teams can transform control in software development and operations. It underscores the influence of shared domain knowledge in achieving alignment between software development and software operations. This study proposes a bivariate management model for aligning cross-functional teams with shared domain knowledge as a critical enabler for the two core control mechanisms controlled-by-competency and controlled-by-invisibility. This dissertation thus contributes to IS research into control element design by proposing a bivariate management model for DevOps teams embedded in the IT function. The results depict the significance of human-to-human control relations as well as human-to-technology relationship in making a transformation toward product orientation in DevOps teams possible. Furthermore, bivariate management lays the groundwork for eliminating conflict potential at an early stage through collaborative communication and a result-driven environment. Next, this research offers insights into how architectural control constrains or enables DevOps teams by examining how technology is used to control work. This research has its roots in 'formal and informal' and 'infrastructure and system' control and builds upon these by demonstrating how they interact with each other.

2.1.3 Intra-IT Alignment

Extant literature has primarily focused on business-IT alignment from a strategic perspective (Gerow et al., 2014b). Nowadays, successful IT functions integrate cross-functional teams in response to shorter deployment cycles for new software development, novel customer requirements and an increasingly complex IT architecture (Fitzgerald & Stol, 2017). These demands underscore the need to align IT development and operation units, which are traditionally organized in separate IT subunits. These units typically pursue different goals and are often misaligned (Henderson and Venkatraman, 1993; Markus and Keil, 1994). Including operations into cross-functional agile software development teams requires internal alignment within DevOps teams. This dissertation describes a path toward aligning operationally software development and software operations activities in DevOps teams to encourage alignment within the IT function.

Building upon operational and intra-IT alignment literature (Onita & Dhaliwal, 2011; Wagner et al., 2014), Paper 8 derives the new tripartite intra-IT alignment model. Three mechanisms serve as the foundation for achieving alignment in DevOps teams. The identified mechanisms explain in the components that connect the business, development, and operations unit perspective.

This research contributes to alignment within the IT function by offering a deeper understanding of the management of software engineering tasks through simultaneously managing software development and operations activities. On one hand, intra-IT alignment confirms that it is supported by setting up DevOps teams. On the other hand, fundamental differences between development and operations arise by including both operations and development topics to encourage toward the transformation of new approaches for IT solutions. Tripartite intra-IT alignment builds upon intra-IT alignment (Dhaliwal et al., 2011; Onita & Dhaliwal, 2011) as well as the SAM by concentrating on operational alignment (Henderson & Venkatraman, 1993).

As outlined above, Paper 7 of this dissertation contributes to control but also to alignment theory. This research concentrates on how shared domain knowledge fosters alignment within DevOps teams. IT alignment research provides insights into the context, specifically shared domain knowledge, that shapes the objectives of the IT function with the objective of the broader organization (Carter et al., 2011; Reich & Benbasat, 2000). Building on scholar's identification of shared domain knowledge as an antecedent of alignment (Reich & Benbasat, 2000) this dissertation demonstrates that shared domain knowledge is required for enabling control and alignment within DevOps teams.

2.2 Implication to Practice

This dissertation has several implications. In this section recommendations for practice to engage with DevOps principles and implementation are provided.

2.2.1 Mitigating Challenges of DevOps Implementation

By reviewing extant literature and the conduction of multiple case studies, challenges of DevOps implementation are systematically identified and recommendations are given on how to respond to these challenges.

First, this dissertation identifies four types of challenges associated with DevOps implementation and proposes that they be considered, managed and resolved from a controller perspective, providing guidelines for controllers derived by equally combining control modes from software development and software operations in Paper 9. By applying the proposed resolutions, the controller can integrate DevOps arrangements that facilitate a productive workable team environment despite the complexity of DevOps transformations.

Second, DevOps implementation has practical challenges, but also opportunities. The technology aspects, such as integrating tool chains for automation, is challenging for incumbent firms used to working with monolithic legacy systems (Gregory et al., 2018). This research shows that the cultural changes associated with the transformation to DevOps team structure is an essential contribution and similar difficult to implement than a suitable tool chain. Strong, transformational leadership that inspires and motivates cultural change is required to rise to this challenge. This research finds that DevOps integration success depends strongly on the context of a firm, and that there is no ‘one size fits all’ solution to challenges, which require individual customized solutions. Hence, adopting DevOps principles should include development of not only technical, but also cultural, process, and measurement practices that are unique in each organization. This requires careful evaluation of the current situation, the specific problems and any barriers to collaboration within the organization to find the most appropriate resolutions.

Third, as part of the digital transformation, more and more organizations are adopting agile or DevOps-oriented IT teams (Sebastian et al., 2017). This thesis provides guidelines for CIOs and IT managers how they could consider the challenges and interdependencies comprehensively to increase the chances of successful DevOps implementation. The organizations should build on their strengths and be aware of their specific weaknesses, while paying close attention to the cultural aspects of the transformation.

2.2.2 Developing Highly Skilled DevOps Team Members

This dissertation presents core insights for training and developing DevOps team members. The findings show that IT managers need to be aware that successful DevOps transformation requires targeted cross-functional training and knowledge sharing. Paper 4 of this dissertation details the comprehensive skill set needed by DevOps teams responsible for one or more IT services. This dissertation finds that successful DevOps implementation requires a holistic view of automation tools and architecture, culture, and processes. DevOps teams must understand how their work affects the organization in more than technical ways. Since many IT employees have highly specialist knowledge in one area (Fitzgerald & Stol, 2014), it is necessary to create awareness of the need for and enable a mindset change. In DevOps teams, a small team is responsible for the complete IT software delivery lifecycle. Hence, team members must be able to substitute for and support each other and take on greater responsibility, which requires a much broader capability and skill set and acquire more generalist knowledge. In order to facilitate DevOps team members’ holistic development, they should undergo training and attend conferences.

2.2.3 Movement to Product Orientation

This dissertation provides managerial implications for moving from project management to product orientation, providing guidelines for managing DevOps teams from a formal and informal control perspective. Along with the shift from project to product orientation, the nature of project management roles changes and a team-member oriented perspective should be taken. IS projects have pre-defined project goals, such as time and budget restrictions (Kirsch, 2004). As a product orientation is adopted, the product team has no starting and ending date. DevOps management continues to have budget constraints, but DevOps teams are ongoing and do not end with a major release or project completion. The transformation to cross-functional IT teams therefore requires a redesign of the processes including end-to-end IT service responsibility (Balaji et al., 2011). This research presents guidelines for managers on how to handle the DevOps transformation, including starting points and key patterns for DevOps, and outlines a control model centered on product management which can be adapted to manage novel product management circumstances.

2.3 Limitations

Since my efforts are directed towards continuous optimization, a constantly advancing research is of critical attention and urgency.

First, the literature reviews within this dissertation consider mainly articles published in the Senior Scholars' Basket of Journals, which were identified as most relevant for this study. Despite forward and backward searches (Webster & Watson, 2002), important relevant contributions in other outlets or in other research areas may not have been considered. Furthermore, the search of this study is limited to findings published since the introduction of the agile manifesto in 2001 (Vial & Rivard, 2015) excluding earlier publications. Despite the attempts to mitigate this limitation through forward and backward searches, which may have inadvertently not considered earlier relevant findings.

Second, this research includes the finding from a workshop aimed to identify a key skill set for DevOps teams with IT consultants with a limited number of participants from one company. The participating persons in the workshops are all IT experts with several years of working experiences and the company is an industry leader in IT implementation projects and consultancy founded more than 30 years ago. The limited scope of participants might limit the generalizability of the research findings. In addition, since the participants were IT consultants and described consultant-client relationships, the view of clients working in internal DevOps teams might be underrepresented.

Third, the global multiple case study includes cases from Europe, America, Australia, Africa, and Asia. However, a high share of the data was gathered from German cases. Despite high efforts in theoretical sampling (Eisenhardt, 1989; Urquhart, 2012) and to win new participants for the study (Yin, 2018), the scope of the research sites might influence the generalizability of the findings. Although similar context factors make the findings more readily comparable so that theoretical conclusions can be drawn, research design mitigates the potential for theoretical statements to be broadened to a larger population (Lee & Baskerville, 2003). The complete data collection process was limited to a time frame of two to three years and the case study participants were in different stages of DevOps implementation. Hence, the comparability of the case study participants might be limited. In this research, mainly experts from DevOps IT teams were considered. Cross-functional teams sometimes have integrated business partners, e.g., as product owners in their teams. Several product owners participated in this study, but all were organizationally assigned to IT, so the business perspective may be underrepresented in this research.

Forth, the main research results of this study are based on qualitative research methodologies. DevOps is a relatively new concept that has gained popularity during the last decade (Reed, 2014). Since the main focus of this inquiry was to investigate incumbent firms during their DevOps transformation, a limited number of companies as case study participants were examined (Yin, 2018). Quantitative research would be valuable to validate the insights derived by this dissertation.

Fifth, the findings of this dissertation mainly concentrate on internal DevOps teams. To be more concrete, the investigation focuses on the setup of internal DevOps team integrated into a company. Outsourcing arrangements such as offshoring relationships are excluded in this dissertation.

2.4 Further Research Opportunities

This dissertation points to several avenues for further research. Generally, further research can verify and build on the concepts presented in this study and provide additional insights into the processes and outcomes of DevOps transformation. The following section explains how the specific underlying theories and novel concepts used in this dissertation can be further developed.

2.4.1 Further Research on IS Capabilities and Skills

The research conducted in this dissertation focus on the concept of skills. As presented in Paper 4, the focus laid on attribute-based competencies concentrating on skill-as-attributes. Skill-as-attributes concentrates on the know-what (Klendauer et al., 2012; Napier et al., 2009) of people. Hence, this thesis provides insights into the requirement for successful DevOps teams. The concept of skill theory also contributes insights into how skills can be developed and how they influence team members. Further research should consider how these skills affect the performance of DevOps teams. The skillful-practice method addresses the behaviors, actions, and activities expected by the team members (Crawford, 2005). Scholars could consider this for future investigation and find out how DevOps teams can be trained and developed.

The papers included in this research focus on the core capabilities and skills that need to be integrated to transform effective DevOps teams. Paper 4 and 5 present a general overview of core insights on what skills should be integrated within a DevOps team and how the planning process is effective. Nevertheless, this dissertation does not provide much information on the relative importance of various skills. Future research should investigate and categorize the skills according to level of importance. Furthermore, considering other sourcing options such as outsourcing arrangement could also lead to new competency models.

2.4.2 Further Research in the Area of Control

This dissertation contains several papers with investigations rooted in control theory from software development as well as software operations perspective. Existing research predominantly focuses on either development (Wiener et al., 2016) or operations control (Benaroch & Chernobai, 2017). This dissertation provides first avenues on how to combine these different control modes and provides novel models with control mechanisms dedicated to DevOps. Five resolutions were identified:

- Foster convergence through technology
- Identify dependencies and differences

- Give authority to guarantor
- Broaden knowledge and skills to higher generalizability
- Enhance end-to-end thinking through cross-functionality

Paper 7 depicts new findings into DevOps control. Digital infrastructure and platform literature calls for further investigations of the impact of IT architecture control so-called non-overt control (Tiwana et al., 2013). Following this call, especially the human-to-technology relationships found in this study deserves further investigations. Integrating operations activities into agile cross-functional teams is supported by new technology like cloud platforms and container tools. For example, as IS architectures grow toward more scalable and interoperable systems:

- How will global connected technological and architectural changes affect DevOps control?
- How will the various use of hardware and software configurations influence the way DevOps teams work?
- How will the opening or closing of digital borders for new technologies or regulations, such as the General Data Protection Regulation change DevOps members' ability to collect, share and integrate information?

Scholars should investigate suitable configurations of control elements in response to unforeseen challenges and changes.

Future research should build upon the findings of Paper 9, which expands existing control modes of software project control (Wiener et al., 2016) as well as software operations control (Edberg et al., 2012) and illustrates how controller and controlees can interact. Future investigations may detail the triggers of choosing control mechanisms. Even though this research presents initial evidence on how control is implemented into incumbent firms, further amplification is needed. Paper 9 focuses on control and relationships, considering DevOps teams as product management-oriented. Scholars could examine suitable outcomes for measuring the successes and failures of DevOps teams and their influence on control. In addition, these findings could be generalized in other context as well (Eisenhardt, 1989).

2.4.3 Further Research in the Area of Alignment

Another research avenue of this dissertation was guided by alignment theory (Henderson & Venkatraman, 1993). In line with the suggestions of Paper 7 and 8, further research may support the process of achieving alignment within the IT function (Onita & Dhaliwal, 2011). Such research can take qualitative and quantitative research approaches to refine the tripartite intra-IT alignment model proposed in Paper 8. In addition, such research could provide tools for measuring intra-IT alignment components. This may increase the understanding how agility can be measured beyond software development as well as the effects on alignment in DevOps teams. Furthermore, the inquiry presented in this study mainly focus on operational alignment dimensions, so complementary research could focus on other alignment dimensions, such as how tripartite intra-IT alignment affects the strategic alignment of business and IT. Furthermore, there could be a potential benefit of an in-depth analysis of a specific instance of

DevOps alignment, especially of longitudinal research based on deep and trusting relationships. Scholars should build upon the findings using data generated by other qualitative research methods, such as observation validated with quantitative methods.

Paper 7 investigates the control-alignment relationships fostered by shared domain knowledge. Extant investigations discover control and alignment in terms of 'peripheral knowledge', which refers to knowledge outside of the specialized domain of activity (Tiwana, 2012; Tiwana & Keil, 2007). Nevertheless, research on shared domain knowledge within IT subunits is rare, (Edberg et al., 2012). Paper 7 provides first insights and derives the new model of bivariate management. Further amplification of this theoretical approach is needed, particularly with regard to how shared domain knowledge is enacted and what factors influence the achievement of high-level shared domain knowledge. Existing research gives first insights that shared domain knowledge can lead to long-term term alignment (Reich & Benbasat, 2000). Scholars could build upon this knowledge and examine how cross-functional teams are aligned from a social alignment perspective.

2.4.4 Further Research in Combination with Meta-Theoretical Lenses

As outlined above, this dissertation contributes to different theories in various forms. Nevertheless, during the research process, it became obvious that further research could combine the findings of this dissertation with other theoretical lenses that are neglected in these studies. To define novel contributions and achieve higher generalizability, the punctuated equilibrium theory (Gersick, 1991; Gregory et al., 2018) could provide valuable insights.

This dissertation builds inter alia up-on control and alignment theory. Existing research highlighted that changes in organizations are ongoing and alignment of IT and business functions must constantly adjusted and renewed (Guillemette & Pare, 2012; Sabherwal et al., 2001). This is also the case in investigations of IT governance and control environments (Gregory et al., 2018).

The transformations from traditional IT functions toward DevOps oriented teams happened in relatively short periods in the investigated cases. Hence, the transformation toward DevOps in the cases followed a "discontinuous, fast, and systemic" (Besson & Rowe, 2012, p. 104) approach, which can be seen as punctuated compared to the original equilibrium period of the traditional and well-established IT functions with silo IT units in the cases. Further research could examine how and why DevOps transforms control and alignment perspectives in large organizations through the lens of punctuated equilibrium theory. Punctuated equilibrium consists of three main concepts: deep structure, equilibrium periods, and revolutionary periods (Gersick, 1991; Sabherwal et al., 2001). Scholars could build upon the findings presented in this dissertation and provide a meta-theoretical approach by incorporating the concepts of punctuated equilibrium.

3. Conclusion

The overall objective of this dissertation is to strengthen the understanding of how organizations can react to, manage, and govern challenges associated with DevOps implementation from a research as well as practitioners' perspective. To do so, several fundamental theories were applied, such as concept of skills, control theory, and business-IT alignment theory, which are also extended in this thesis. This dissertation provides empirical evidence and theoretical models that empower organizations to implement DevOps teams. The theoretical basis underlying this research offers new insights for research and enable scholars for further research opportunities.

The main research results are based on different research methodologies. The main part of the findings is based on evidence from case study investigations undertaken in Germany but also with multinational participants located in America, Australia, Africa, and Asia. Building upon IS capabilities and skills, this dissertation offers an overview of possible DevOps setups and how this concept can be integrated by incumbent firms. In addition, this thesis finds that control mechanisms from software development and software operations must be combined to manage DevOps teams effectively and efficiently. Furthermore, new control mechanisms are needed to enable DevOps teams. This dissertation develops a bivariate management model illustrated by a control-alignment DevOps tetrahedron. Drawing on alignment literature, this thesis contributes to research in the area of operational business-IT alignment, especially alignment within the IT function. In effective DevOps teams, agility is broadened to software operations and accelerates alignment within the IT function. The tripartite intra-IT alignment model developed in this thesis provides insights into this topic.

This dissertation was guided by three overall research questions that were analyzed with the help of nine research papers. Existing IS research is extended by providing a better understanding for scholars and showing unique contributions by investigation on cross-functional teams. Furthermore, this thesis adds value to practice by proposing guidelines for managing DevOps teams from the executive as well as the team-level perspective.

Finally, this dissertation contributes to research providing a comprehensive overview of managing organizational, cultural, technological, and human challenges during digital transformation by integrating DevOps teams within existing IT functions.

References

Aanestad, M., & Jensen, T. B. (2011). Building Nation-Wide Information Infrastructures in Healthcare through Modular Implementation Strategies. *The Journal of Strategic Information Systems, 20*(2), 161-176.

Addy, R. (2007). *Effective IT Service Management: To ITIL and Beyond!* New York, USA: Springer, Inc.

Advance IT Minnesota. (2016). *Renewing the IT Curriculum: Responding to Agile, DevOps, and Digital Transformation*. Retrieved from St. Paul, MN: www.DynamicIT.education

Ågerfalk, P. J., Fitzgerald, B., & Slaughter, S. A. (2009). Introduction to the Special Issue-Flexible and Distributed Information Systems Development: State of the Art and Research Challenges. *Information Systems Research, 20*(3), 317-328.

Alter, S. (2008). Defining Information Systems as Work Systems: Implications for the IS Field. *European Journal of Information Systems, 17*(5), 448-469.

Alter, S. (2013). Work System Theory: Overview of Core Concepts, Extensions, and Challenges for the Future. *Journal of the Association for Information Systems, 14*(2), 72.

April, A., Huffman Hayes, J., Abran, A., & Dumke, R. (2005). Software Maintenance Maturity Model (SMmm): The Software Maintenance Process Model. *Journal of Software Maintenance and Evolution: Research and Practice, 17*(3), 197-223.

Arvidsson, V., Holmström, J., & Lyytinen, K. (2014). Information Systems Use as Strategy Practice: A Multi-Dimensional View of Strategic Information System Implementation and Use. *The Journal of Strategic Information Systems, 23*(1), 45-61.

Avison, D. E., & Fitzgerald, G. (2003). Where Now for Development Methodologies? *Communications of the ACM, 46*(1), 78-82.

Balaji, S., Ranganathan, C., & Coleman, T. (2011). IT-Led Process Reengineering: How Sloan Valve Redesigned its New Product Development Process. *MIS Quarterly Executive, 10*(2), 81-92.

Balijepally, V., Mahapatra, R., Nerur, S., & Price, K. H. (2009). Are Two Heads Better than One for Software Development? The Productivity Paradox of Pair Programming. *MIS Quarterly, 33*(1), 91-118.

Bang, S. K., Chung, S., Choh, Y., & Dupuis, M. (2013). *A Grounded Theory Analysis of Modern Web Applications: Knowledge, Skills, and Abilities for DevOps*. Paper presented at the Proceedings of the 2nd Annual Conference on Research in Information Technology, Orlando, USA.

Banker, R. D., Datar, S. M., & Kemerer, C. F. (1991). A Model to Evaluate Variables Impacting the Productivity of Software Maintenance Projects. *Management Science, 37*(1), 1-18.

Banker, R. D., Datar, S. M., Kemerer, C. F., & Zweig, D. (2002). Software Errors and Software Maintenance Management. *Information Technology and Management, 3*(1-2), 25-41.

Banker, R. D., Davis, G. B., & Slaughter, S. A. (1998). Software Development Practices, Software Complexity, and Software Maintenance Performance: A Field Study. *Management Science, 44*(4), 433-450.

Banker, R. D., Slaughter, S., Swatman, P., Wagenaar, R., & Wrigley, C. (1994). *Project Size and Software Maintenance Productivity: Empirical Evidence on Economies of Scale in Software Maintenance*. Paper presented at the International Conference on Information Systems, New York, USA.

Bansal, P., & Corley, K. (2012). Publishing in AMJ—Part 7: What's Different about Qualitative Research? *Academy of Management Journal, 55*(3), 509-513.

Bardhan, I. R., Demirkan, H., Kannan, P. K., Kauffman, R. J., & Sougstad, R. (2010). An Interdisciplinary Perspective on IT Services Management and Service Science. *Journal of Management Information Systems, 26*(4), 13-64.

Baronas, A.-M. K., & Louis, M. R. (1988). Restoring a Sense of Control During Implementation: How User Involvement Leads to System Acceptance. *MIS Quarterly*, *12*(1), 111-124.

Barua, A., Konana, P., Whinston, A. B., & Yin, F. (2004). An Empirical Investigation of Net-Enabled Business Value. *MIS Quarterly, 28*(4), 585-620.

Baskerville, R., & Pries-Heje, J. (2004). Short Cycle Time Systems Development. *Information Systems Journal, 14*(3), 237-264.

Bass, L., Weber, I., & Liming, Z. (2015). *DevOps: A Software Architect's Perspective*. Old Tappan, New Jersey: Pearson Education, Inc.

Bassellier, G., Benbasat, I., & Reich, B. H. (2003). The Influence of Business Managers' IT Competence on Championing IT. *Information Systems Research, 14*(4), 317-336.

Baumgart, R., Hummel, M., & Holten, R. (2015). *Personality Traits of Scrum Roles in Agile Software Development Teams- A Qualitative Analysis*. Paper presented at the European Conference on Information Systems, Münster, Germany.

Beck, K., Beedle, M., Bennekum, v. A., Cockburn, A., Cunningham, W., Fowler, M., . . . Thomas, D. (2001). Manifesto for Agile Software Development. http://agilemanifesto.org

Benaroch, M., & Chernobai, A. (2017). Operational IT Failures, IT Falue-Destruction, and Board-Level IT Governance Changes. *MIS Quarterly, 41*(3).

Besson, P., & Rowe, F. (2012). Strategizing Information Systems-Enabled Organizational Transformation: A Transdisciplinary Review and New Directions. *The Journal of Strategic Information Systems, 21*(2), 103-124.

Bharadwaj, A., El Sawy, O. A., Pavlou, P. A., & Venkatraman, N. (2013). Digital Business Strategy: Toward a Next Generation of Insights. *MIS Quarterly, 37*(2), 471-482.

Bharadwaj, A., Keil, M., & Mähring, M. (2009). Effects of Information Technology Failures on the Market Value of Firms. *The Journal of Strategic Information Systems, 18*(2), 66-79.

Bharadwaj, A. S. (2000). A Resource-Based Perspective on Information Technology Capability and Firm Performance: an Empirical Investigation. *MIS Quarterly, 24*(1), 169-196.

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.

Bhatt, G., Emdad, A., Roberts, N., & Grover, V. (2010). Building and Leveraging Information in Dynamic Environments: The Role of IT Infrastructure Flexibility as Enabler of Organizational Responsiveness and Competitive Advantage. *Information & Management, 47*(7), 341-349.

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.

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.

Blohm, I., Bretschneider, U., Leimeister, J. M., & Krcmar, H. (2010). *Does Collaboration Among Participants Lead to Better Ideas in IT-based Idea Competitions? An Empirical Investigation*. Paper presented at the Hawaii International Conference on System Sciences, Kauai, USA.

Boehm, M., Stolze, C., & Thomas, O. (2013). *Teaching the Chief Information Officers: An Assessment of the Interrelations within their Skill Set*. Paper presented at the Wirtschaftsinformatik, Leipzig, Germany.

Bogner, A., Littig, B., & Menz, W. (2009). Introduction: Expert Interviews—An Introduction to a New Methodological Debate. In *Interviewing Experts* (pp. 1-13): Springer.

Boonstra, A., Yeliz Eseryel, U., & van Offenbeek, M. A. G. (2017). Stakeholders' Enactment of Competing Logics in IT Governance: Polarization, Compromise or Synthesis? *European Journal of Information Systems*, 1-19.

Botzenhardt, A., Meth, H., & Maedche, A. (2011). *Cross-Functional Integration of Product Management and Product Design in Application Software Development: Exploration of Success Factors*. Paper presented at the International Conference on Information Systems, Shanghai, China.

Broadbent, M., Weill, P., & St. Clair, D. (1999). The Implications of Information Technology Infrastructure for Business Process Redesign. *MIS Quarterly, 23*(2), 159-182.

Brown, A., Forsgren, N., Humble, J., Kersten, N., & Kim, G. (2016). *State of DevOps Report Puppet + DORA.*

Brown, C. V., & Ross, J. W. (1996). The Information Systems Balancing Act: Building Partnerships and Infrastructure. *Information Technology & People, 9*(1), 49-62.

Burke, C. S., Stagl, K. C., Salas, E., Pierce, L., & Kendall, D. (2006). Understanding Team Adaptation: A Conceptual Analysis and Model. *Journal of Applied Psychology, 91*(6), 1189-1207.

Bush, A. A., Tiwana, A., & Rai, A. (2010). Complementarities between Product Design Modularity and IT Infrastructure Flexibility in IT-enabled Supply Chains. *IEEE Transactions on Engineering Management, 57*(2), 240-254.

Cao, L., Mohan, K., Ramesh, B., & Sarkar, S. (2013). Adapting Funding Processes for Agile IT Projects: An Empirical Investigation. *European Journal of Information Systems, 22*(2), 191-205.

Cao, L., Mohan, K., Xu, P., & Ramesh, B. (2009). A Framework for Adapting Agile Development Methodologies. *European Journal of Information Systems, 18*(4), 332-343.

Carter, M., Grover, V., & Thatcher, J. B. (2011). The Emerging CIO Role of Business Technology Strategist. *MIS Quarterly Executive, 10*(1), 19-29.

Cater-Steel, A., & Tan, W.-G. (2005). *Implementation of IT Infrastructure Library (ITIL) in Australia: Progress and Success Factors.* Paper presented at the IT Governance International Conference.

Chakraborty, S., Sarker, S., & Sarker, S. (2010). An Exploration into the Process of Requirements Elicitation: A Grounded Approach. *Journal of the Association for Information Systems, 11*(4), 212.

Châlons, C., & Dufft, N. (2017). The Role of IT as an Enabler of Digital Transformation. In F. Abolhassan (Ed.), *The Drivers of Digital Transformation* (pp. 13-22). Saarbrücken, Germany Springer.

Chan, Y. E., & Reich, B. H. (2007). IT Alignment: What Have we Learned? *Journal of Information Technology, 22*(4), 297-315.

Charmaz, K. (2000). Grounded Theory: Objectivist and Constructivist Methods. *Handbook of Qualitative Research, 2*, 509-535.

Chen, D. Q., Mocker, M., Preston, D. S., & Teubner, A. (2010). Information Systems Strategy: Reconceptualization, Measurement, and Implications. *MIS Quarterly, 34*(2), 233-259.

Chen, L. (2017). Continuous Delivery: Overcoming Adoption Challenges. *The Journal of Systems and Software, 128*, 72–86.

Choi, J., Nazareth, D. L., & Jain, H. K. (2010). Implementing Service-Oriented Architecture in Organizations. *Journal of Management Information Systems, 26*(4), 253-286.

Choudhury, V., & Sabherwal, R. (2003). Portfolios of Control in Outsourced Software Development Projects. *Information Systems Research, 14*(3), 291-314.

Chow, T., & Cao, D.-B. (2008). A Survey Study of Critical Success Factors in Agile Software Projects. *Journal of Systems and Software, 81*(6), 961-971.

Christopher, M., & Towill, D. R. (2000). Supply Chain Migration from Lean and Functional to Agile and Customised. *Supply Chain Management: An International Journal, 5*(4), 206-213.

Chua, C. E. H., Lim, W.-K., Soh, C., & Sia, S. K. (2012). Enacting Clan Control in Complex IT Projects: A Social Capital Perspective. *MIS Quarterly, 36*(2), 577-600.

Conboy, K. (2011). *Comparing Apples with Oranges? The Perceived Differences between Agile and Lean Software Development Processes*. Paper presented at the International Conference on Information Systems Shanghai, China.

Constantinides, P., & Barrett, M. (2014). Information Infrastructure Development and Governance as Collective Action. *Information Systems Research, 26*(1), 40-56.

Coyle, S., Conboy, K., & Acton, T. (2015). *An Exploration of the Relationship Between Contribution Behaviours and the Decision Making Process in Agile Teams*. Paper presented at the International Conference on Information Systems, Fort Worth, USA.

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.

Cram, W. A., Brohman, K., & Gallupe, R. B. (2016a). Information Systems Control: A Review and Framework for Emerging Information Systems Processes. *Journal of the Association for Information Systems, 17*(4), 216-266.

Cram, W. A., & Brohman, M. K. (2013). Controlling Information Systems Development: A New Typology for an Evolving Field. *Information Systems Journal, 23*(2), 137-154.

Cram, W. A., Brohman, M. K., Chan, Y. E., & Gallupe, R. B. (2016b). Information Systems Control Alignment: Complementary and Conflicting Systems Development Controls. *Information and Management, 53*(2), 183-196.

Crawford, L. (2005). Senior Management Perceptions of Project Management Competence. *International Journal of Project Management, 23*(1), 7-16.

Crossan, M. M., Lane, H. W., & White, R. E. (1999). An Organizational Learning Framework: From Intuition to Institution. *Academy of Management Review, 24*(3), 522-537.

De Haes, S., & Van Grembergen, W. (2004). IT Governance and its Mechanisms. *Information Systems Control Journal, 1*, 27-33.

Debois, P. (2010). Skills Matter - Inside the Brain of Patrick Debois - Jumpstarting a Devops Mentality In *Jedi.be*.

Debois, P. (2011). Opening Statement. *Cutter IT Journal, 24*(8), 3-5.

Dekleva, S. M. (1992). The Influence of the Information Systems Development Approach on Maintenance. *MIS Quarterly, 16*(3), 355-372.

Dennis, A. R., Samuel, B. M., & McNamara, K. (2014). Design for Maintenance: how KMS Document Linking Decisions Affect Maintenance Effort and Use. *Journal of Information Technology, 29*(4), 312-326.

Dery, K., Sebastian, I. M., & van der Meulen, N. (2017). The Digital Workplace is Key to Digital Innovation. *MIS Quarterly Executive, 16*(2), 135-152.

Devopsdays. (2009). Devopsdays Ghent 2009 Retrieved from https://legacy.devopsdays.org/events/2009-ghent/

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.

Dougherty, D. (1992). Interpretive Barriers to Successful Product Innovation in Large Firms. *Organization Science, 3*(2), 179-202.

Duh, R.-R., Chow, C. W., & Chen, H. (2006). Strategy, IT Applications for Planning and Control, and Firm Performance: The Impact of Impediments to IT Implementation. *Information & Management, 43*(8), 939-949.

Dybå, T., & Dingsøyr, T. (2008). Empirical Studies of Agile Software Development: A Systematic Review. *Information and Software Technology, 50*(9), 833-859.

Ebert, C., Gallardo, G., Hernantes, J., & Serrano, N. (2016). DevOps. *IEEE Software, 33*(3), 94-100.

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.

Eisenhardt, K. M. (1985). Control: Organizational and Economic Approaches. *Management Science, 31*(2), 134-149.

Eisenhardt, K. M. (1989). Building Theories from Case Study Research. *Academy of Management Review, 14*(4), 532-550.

Eseryel, U. Y., & Eseryel, D. (2013). Action-Embedded Transformational Leadership in Self-Managing Global Information Systems Development Teams. *The Journal of Strategic Information Systems, 22*(2), 103-120.

Fang, C., Lee, J., & Schilling, M. A. (2010). Balancing Exploration and Exploitation through Structural Design: The Isolation of Subgroups and Organizational Learning. *Organization Science, 21*(3), 625-642.

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.

Fielt, E., Böhmann, T., Korthaus, A., Conger, S., & Gable, G. (2013). Service Management and Engineering in Information Systems Research. *The Journal of Strategic Information Systems, 22*(1), 46-50.

Fink, L., & Neumann, S. (2007). Gaining Agility through IT Personnel Capabilities: The Mediating Role of IT Infrastructure Capabilities. *Journal of the Association for Information Systems, 8*(8), 440-462.

Fiol, C. M., & Lyles, M. A. (1985). Organizational Learning. *Academy of Management Review, 10*(4), 803-813.

Fitzgerald, B. (1998). An Empirical Investigation into the Adoption of Systems Development Methodologies. *Information & Management, 34*(6), 317-328.

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.

Fitzgerald, B., & Stol, K.-J. (2014). Continuous Software Engineering and Beyond: Trends and Challenges. In *International Workshop on Rapid Continuous Software Engineering* (pp. 1-9): ACM.

Fitzgerald, B., & Stol, K.-J. (2017). Continuous Software Engineering: A Roadmap and Agenda. *Journal of Systems and Software, 123*, 176–189.

Fladmoe-Lindquist, K., & Jacque, L. L. (1995). Control Modes in International Service Operations: The Propensity to Franchise. *Management Science, 41*(7), 1238-1249.

Flyvbjerg, B. (2006). Five Misunderstandings About Case-Study Research. *Qualitative Inquiry, 12*(2), 219-245.

Forsgren, N. (2018). DevOps Delivers. *Communications of the ACM, 61*(4), 32-33.

Forsgren, N., Durcikova, A., Clay, P. F., & Wang, X. (2016). The Integrated User Satisfaction Model: Assessing Information Quality and System Quality as Second-order Constructs in System Administration. *Communications of the Association for Information Systems, 38*, 803-839.

Forsgren, N., & Humble, J. (2015). *The Role of Continuous Delivery in IT and Organizational Performance*. Paper presented at the Western Decision Sciences Institute Las Vegas, USA.

Forsgren, N., & Humble, J. (2016). *DevOps: Profiles in ITSM Performance and Contributing Factors*. Paper presented at the Western Decision Sciences Institute, Las Vegas, USA.

Forsgren, N., Humble, J., & Kim, G. (2018). *The Science Behind DevOps: Accelerate Building and Scaling High Performing Technology Organizations*. Portland, Oregon: IT Revolution.

Forsgren, N., Humble, J., Kim, G., Brown, A., & Kersten, N. (2017). *State of DevOps Report*: puppet+DORA.

Forsgren, N., Humble, J., Kim, G., Brown, A., & Kersten, N. (2018a). *Accelerate: State of DevOps Strategies for a New Economy*. DORA+Google Cloud.

Forsgren, N., & Kersten, M. (2018). DevOps Metrics. *Communications of the ACM, 61*(4), 44-48.

Forsgren, N., Sabherwal, R., & Durcikova, A. (2018b). Knowledge Exchange Roles and EKR Performance Impact: Extending the Theory of Knowledge Reuse. *European Journal of Information Systems, 27*(1), 3-21.

Fruhling, A., & Vreede, G.-J. D. (2006). Field Experiences with eXtreme Programming: Developing an Emergency Response System. *Journal of Management Information Systems, 22*(4), 39-68.

Garousi, V., Felderer, M., & Mäntylä, M. V. (2016). *The Need for Multivocal Literature Reviews in Software Engineering*. Paper presented at the International Conference on Evaluation and Assessment in Software Engineering, Limerick, Ireland.

Gerow, J. E., Grover, V., Thatcher, J. B., & Roth, P. L. (2014a). Looking Toward the Future of IT-Business Strategic Alignment through the Past: A Meta-Analysis. *MIS Quarterly, 38*(4), 1059-1085.

Gerow, J. E., Thatcher, J. B., & Grover, V. (2014b). Six Types of IT-Business Strategic Alignment: An Investigation of the Constructs and their Measurement. *European Journal of Information Systems, 24*(5), 465-491.

Gersick, C. J. (1991). Revolutionary Change Theories: A Multilevel Exploration of the Punctuated Equilibrium Paradigm. *Academy of Management Review, 16*(1), 10-36.

Ghazawneh, A., & Henfridsson, O. (2013). Balancing Platform Control and External Contribution in Third-Party Development: The Boundary Resources Model. *Information Systems Journal, 23*(2), 173-192.

Ghobadi, S., & Mathiassen, L. (2016). Perceived Barriers to Effective Knowledge Sharing in Agile Software Teams. *Information Systems Journal, 26*(2), 95-125.

Gibson, C. B., & Birkinshaw, J. (2004). The Antecedents, Consequences, and Mediating Role of Organizational Ambidexterity. *Academy of Management Journal, 47*(2), 209-226.

Glaser, B. G. (1978). *Theoretical Sensitivity: Advances in the Methodology of Grounded Theory*. San Francisco, USA: The Sociology Press.

Glaser, B. G., & Strauss, A. L. (1967). *The Discovery of Grounded Theory: Strategies for Qualitative Research*. New York, NY: Aldine Publishing Company.

Glaser, B. G., & Strauss, A. L. (1998). *Grounded Theory: Strategien Qualitativer Forschung*. Bern: Huber.

Goh, J. C.-L., Pan, S. L., & Zuo, M. (2013). Developing the Agile IS Development Practices in Large-Scale IT Projects: the Trust-Mediated Organizational Controls and IT Project Team Capabilities Perspectives. *Journal of the Association for Information Systems, 14*(12), 722-756.

Goldstein, J., Chernobai, A., & Benaroch, M. (2011). An Event Study Analysis of the Economic Impact of IT Operational Risk and Its Subcategories. *Journal of the Association for Information Systems, 12*(9), 606.

Gordon, J. R., & Gordon, S. R. (2002). Information Technology Service Delivery: An International Comparison. *Information Systems Management, 19*(1), 62-70.

Gotel, O., & Leip, D. (2007). Agile Software Development Meets Corporate Deployment Procedures: Stretching the Agile Envelope. *Agile Processes in Software Engineering and Extreme Programming*, 24-27.

Grant, R. M. (1991). The Resource-based Theory of Competitive Advantage: Implications for Strategy Formulation. *California Management Review, 33*(3), 114-135.

Grant, R. M. (1996). Toward a Knowledge-Based Theory of the Firm. *Strategic Management Journal, 17*(S2), 109-122.

Gregory, R. W., Beck, R., & Keil, M. (2013). Control Balancing in Information Systems Development Offshoring Projects. *MIS Quarterly, 37*(4), 1211-1232.

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.

Guillemette, M. G., & Pare, G. (2012). Toward a New Theory of the Contribution of the IT Function in Organizations. *MIS Quarterly, 36*(2), 529-551.

Gupta, A. K., Smith, K. G., & Shalley, C. E. (2006). The Interplay Between Exploration and Exploitation. *Academy of Management Journal, 49*(4), 693-706.

Haffke, I., Kalgovas, B., & Benlian, A. (2017). Options for Transforming the IT Function Using Bimodal IT. *MIS Quarterly Executive, 16*(2), 101-120.

Hanseth, O., & Monteiro, E. (1997). Inscribing Behaviour in Information Infrastructure Standards. *Accounting, Management Information Technologies, 7*(4), 183-211.

Harris, M. L., Collins, R. W., & Hevner, A. R. (2009). Control of Flexible Software Development Under Uncertainty. *Information Systems Research, 20*(3), 400-419.

He, Z.-L., & Wong, P.-K. (2004). Exploration vs. Exploitation: An Empirical Test of the Ambidexterity Hypothesis. *Organization Science, 15*(4), 481-494.

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.

Hemon, A., & Rowe, F. (2019). *IS Career Anchors, Professional Growth and Mobility Intentions: A DevOps Jobs Learning Effect?* Paper presented at the International Conference on Information Systems, Munich.

Henderson, J. C., & Lee, S. (1992). Managing I/S Design Teams: A Control Theories Perspective. *Management Science, 38*(6), 757-777.

Henderson, J. C., & Venkatraman, N. (1993). Strategic Alignment: Leveraging Information Technology for Transforming Organizations. *IBM Systems Journal, 32*(1), 4-16.

Henfridsson, O., & Bygstad, B. (2013). The Generative Mechanisms of Digital Infrastructure Evolution. *MIS Quarterly, 37*(3), 907-931.

Heumann, J., Wiener, M., Remus, U., & Mähring, M. (2015). To Coerce or to Enable? Exercising Formal Control in a Large Information Systems Project. *Journal of Information Technology, 30*(4), 337-351.

Hipkin, I. (1996). Evaluating Maintenance Management Information Systems. *European Journal of Information Systems, 5*(4), 261-272.

Hoermann, S., Hlavka, T., Schermann, M., & Krcmar, H. (2015). Determinants of Vendor Profitability in Two Contractual Regimes: An Empirical Analysis of Enterprise Resource Planning Projects. *Journal of Information Technology, 30*(4), 325-336.

Holmqvist, M., & Pessi, K. (2006). Agility through Scenario Development and Continuous Implementation: a Global Aftermarket Logistics Case. *European Journal of Information Systems, 15*(2), 146-158.

Horlach, B., Drews, P., Schirmer, I., & Boehmann, T. (2017). *Increasing the Agility of IT Delivery: Five Types of Bimodal IT Organization*. Paper presented at the Hawaii International Conference on System Sciences, Waikaola Village, USA.

Hovorka, D. S., & Larsen, K. R. (2006). Enabling Agile Adoption Practices through Network Organizations. *European Journal of Information Systems, 15*(2), 159-168.

Howard, M., Vidgen, R., & Powell, P. (2004). *Exploring Industry Dynamics in eProcurement: Sense Making by Collaborative Investigation*. Paper presented at the European Conference on Information Systems, Turku, Finland.

Humble, J. (2018). Continuous Delivery Sounds Great, But Will it Work Here? *Communications of the ACM, 61*(4), 34-39.

Humble, J., & Farley, D. (2011). *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*. Boston, MA: Pearson Education, Inc.

Humble, J., & Molesky, J. (2011). Why Enterprises must Adopt DevOps to Enable Continuous Delivery. *Cutter IT Journal, 24*(8), 6-12.

Hummel, M., Rosenkranz, C., & Holten, R. (2013a). *Explaining The Changing Communication Paradigm Of Agile Information Systems Development: A Research Model, Measurement Development And Pretest*. Paper presented at the European Conference on Information Systems, Utrecht, Netherlands.

Hummel, M., Rosenkranz, C., & Holten, R. (2013b). The Role of Communication in Agile Systems Development. *Business & Information Systems Engineering, 5*(5), 343-355.

IBM. (2013). White Paper DevOps: The IBM Approach - Continuous Delivery of Software-Driven Innovation. In *IBM Software*.

IEEE. (1998). *IEEE Standard for Software Maintenance, Std 1219-1998*. Retrieved from New York, USA:

Institute, P. M. (2002). *Project Manager Competency Development Framework*. Newtown Square, USA: Project Management Institute.

ISACA. (2012). *COBIT - 5th Edition*. Information Systems Audit and Control Foundation, Rolling Meadows, USA.

Ives, B., & Vitale, M. R. (1988). After the Sale: Leveraging Maintenance with Information Technology. *MIS Quarterly, 12*(1), 7-21.

Jaworski, B. J. (1988). Toward a Theory of Marketing Control: Environmental Context, Control Types, and Consequences. *The Journal of Marketing, 52*(3), 23-39.

Jia, R., & Reich, B. H. (2013). IT Service Climate, Antecedents and IT Service Quality Outcomes: Some Initial Evidence. *The Journal of Strategic Information Systems, 22*(1), 51-69.

Jöhnk, J., Röglinger, M., Thimmel, M., & Urbach, N. (2017). *How to Implement Agile IT Setups: A Taxonomy of Design Options*. Paper presented at the European Conference on Information Systems, Guimarães, Portugal.

Kang, S.-C., Morris, S. S., & Snell, S. A. (2007). Relational Archetypes, Organizational Learning, and Value Creation: Extending the Human Resource Architecture. *Academy of Management Review, 32*(1), 236-256.

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.

Kang, S. C., & Snell, S. A. (2009). Intellectual Capital Architectures and Ambidextrous Learning: a Framework for Human Resource Management. *Journal of Management Studies, 46*(1), 65-92.

Kaplan, B., & Maxwell, J. A. (2005). Qualitative Research Methods for Evaluating Computer Information Systems. In *Evaluating the Organizational Impact of Healthcare Information Systems* (pp. 30-55): Springer.

Karahanna, E., & Preston, D. S. (2013). The Effect of Social Capital of the Relationship between the CIO and Top Management Team on Firm Performance. *Journal of Management Information Systems, 30*(1), 15-56.

Keil, M., Lee, H. K., & Deng, T. (2013). Understanding the Most Critical Skills for Managing IT Projects: A Delphi Study of IT Project Managers. *Information & Management, 50*(7), 398-414.

Kemerer, C. F., & Slaughter, S. (1999). An Empirical Approach to Studying Software Evolution. *IEEE Transactions on Software Engineering, 25*(4), 493-509.

Kim, C., & Westin, S. (1988). Software Maintainability: Perceptions of EDP Professionals. *MIS Quarterly, 12*(2), 167-185.

Kim, G., Humble, J., Debois, P., & Willis, J. (2016). *The DevOps Handbook*. Portland, USA: IT Revolution Press, LLC.

Kim, S., Park, S., Yun, J., & Lee, Y. (2008). *Automated Continuous Integration of Component-Based Software: An Industrial Experience*. Paper presented at the Proceedings of the 2008 23rd IEEE/ACM International Conference on Automated Software Engineering.

Kirsch, L. J. (1996). The Management of Complex Tasks in Organizations: Controlling the Systems Development Process. *Organization Science, 7*(1), 1-21.

Kirsch, L. J. (2004). Deploying Common Systems Globally: The Dynamics of Control. *Information Systems Research, 15*(4), 374-395.

Kirsch, L. J., Ko, D.-G., & Haney, M. H. (2010). Investigating the Antecedents of Team-Based Clan Control: Adding Social Capital as a Predictor. *Organization Science, 21*(2), 469-489.

Kirsch, L. J., Sambamurthy, V., Ko, D.-G., & Purvis, R. L. (2002). Controlling Information Systems Development Projects: The View from the Client. *Management Science, 48*(4), 484-498.

Kirsch, L. S. (1997). Portfolios of Control Modes and IS Project Management. *Information Systems Research, 8*(3), 215-239.

Klendauer, R., Berkovich, M., Gelvin, R., Leimeister, J. M., & Krcmar, H. (2012). Towards a Competency Model for Requirements Analysts. *Information Systems Journal, 22*(6), 475-503.

Kling, R., & Iacono, S. (1984). The Control of Information Systems Developments after Implementation. *Communications of the ACM, 27*(12), 1218-1226.

Knight, S., Rabideau, G., Chien, S., Engelhardt, B., & Sherwood, R. (2001). Casper: Space Exploration Through Continuous Planning. *IEEE Intelligent Systems, 16*(5), 70-75.

Kohavi, R., Crook, T., Longbotham, R., Frasca, B., Henne, R., Ferres, J. L., & Melamed, T. (2009). Online Experimentation at Microsoft. *Data Mining Case Studies*.

Koutsikouri, D., Lindgren, R., Henfridsson, O., & Rudmark, D. (2018). Extending Digital Infrastructures: A Typology of Growth Tactics. *Journal of the association for information systems, 19*(10), 1001-1019.

Krancher, O., & Luther, P. (2015). *Software Development in the Cloud: Exploring the Affordances of Platform-as-a-Service*. Paper presented at the International Conference on Information Systems, Fort Worth, USA.

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.

Kude, T., Bick, S., Schmidt, C., & Heinzl, A. (2014). *Adaptation Patterns in Agile Information Systems Development Teams*. Paper presented at the European Conference on Information Systems, Tel Aviv, Israel.

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.

Lamb, R., & Kling, R. (2003). Reconceptualizing Users as Social Actors in Information Systems Research. *MIS Quarterly, 27*(2), 197-236.

Laney, D., & Jain, A. (2016). *100 Data and Analytics Predictions Through 2020*. Stamford, USA: Gartner, Inc.

Lassak, S., Przybilla, L., Wiesche, M., & Krcmar, H. (2017). *Explaining How Agile Software Development Practices Moderate the Negative Effects of Faultlines in Teams*. Paper presented at the Australasian Conference on Information Systems, Hobart, Australia.

Lee, A. S., & Baskerville, R. L. (2003). Generalizing Generalizability in Information Systems Research. *Information Systems Research, 14*(3), 221-243.

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.

Lee, O.-K., Sambamurthy, V., Lim, K. H., & Wei, K. K. (2015). How Does IT Ambidexterity Impact Organizational Agility? *Information Systems Research, 26*(2), 398-417.

Lehtola, L., Kauppinen, M., Vähäniitty, J., & Komssi, M. (2009). Linking Business and Requirements Engineering: Is Solution Planning a Missing Activity in Software Product Companies? *Requirements Engineering, 14*(2), 113-128.

Leidner, D. E., Urbach, N., Drews, P., & Ross, J. (2017). Digital Business Transformation and the Changing Role of the IT Function. *MIS Quarterly Executive, 16*(2), ii-iv.

Lempinen, H., & Rajala, R. (2014). Exploring Multi-Actor Value Creation in IT Service Processes. *Journal of Information Technology, 29*(2), 170-185.

Li, Y., & Maedche, A. (2012). *Formulating Effective Coordination Strategies in Agile Global Software Development Teams*. Paper presented at the International Conference on Information Systems, Orlando, USA.

Lin, T.-C., Chen, C.-M., Hsu, J. S.-C., & Fu, T.-W. (2015). The Impact of Team Knowledge on Problem Solving Competence in Information Systems Development Team. *International Journal of Project Management, 33*(8), 1692-1703.

Liu, G. H., Wang, E., & Chua, C. E. H. (2015). Leveraging Social Capital to Obtain Top Management Support in Complex, Cross-Functional IT Projects. *Journal of the Association for Information Systems, 16*(8), 707-737.

Lowry, P. B., & Wilson, D. (2016). Creating Agile Organizations Through IT: The Influence of Internal IT Service Perceptions on IT Service Quality and IT Agility. *The Journal of Strategic Information Systems, 25*(3), 211-226.

Lucia, A. D., & Lepsinger, R. (1999). *Art and Science of Competency Models*. San Francisco, USA: Jossey-Bass

Lwakatare, L. E., Karvonen, T., Sauvola, T., Kuvaja, P., Olsson, H. H., Bosch, J., & Oivo, M. (2016). *Towards DevOps in the Embedded Systems Domain: Why is it so Hard?* Paper presented at the Hawaii International Conference on System Sciences Kauai, USA.

Mähring, M. (2002). IT Project Governance: A Process-Oriented Study of Organizational Control and Executive Involvement. *SSE/EFI Working Paper Series in Business Administration, 15*.

Majchrzak, A., Malhotra, A., & John, R. (2005). Perceived Individual Collaboration Know-How Development through Information Technology–Enabled Contextualization: Evidence from Distributed Teams. *Information Systems Research, 16*(1), 9-27.

Majchrzak, A., More, P. H. B., & Faraj, S. (2012). Transcending Knowledge Differences in Cross-Functional Teams. *Organization Science, 23*(4), 951-970.

Mangalaraj, G., Mahapatra, R., & Nerur, S. (2009). Acceptance of Software Process Innovations – The Case of Extreme Programming. *European Journal of Information Systems, 18*(4), 344-354.

Mansfield, R. S. (1996). Building Competency Models: Approaches for HR Professionals. *Human Resource Management, 35*(1), 7-18.

March, J. D. (1991). Exploration and Exploitation in Organizational Learning. *Organization Science, 2*(1), 71-87.

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.

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*.

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.

Matook, S., & Maruping, L. M. (2014). A Competency Model for Customer Representatives in Agile Software Development Projects. *MIS Quarterly Executive, 13*(2), 77-95.

Matook, S., Soltani, S., & Maruping, L. (2016). *Self-Organization in Agile ISD Teams and the Influence on Exploration and Exploitation*. Paper presented at the International Conference on Information Systems, Dublin, Ireland.

Maxwell, J. A. (2012). *Qualitative Research Design: An Interactive Approach* (Vol. 3). Thousand Oaks: Sage Publications.

McAvoy, J., & Butler, T. (2009). The Role of Project Management in Ineffective Decision Making within Agile Software Development Projects. *European Journal of Information Systems, 18*(4), 372-383.

McAvoy, J., Nagle, T., & Sammon, D. (2013). Using Mindfulness to Examine ISD Agility. *Information Systems Journal, 23*(2), 155-172.

Melnyk, S. A., Stewart, D. M., & Swink, M. (2004). Metrics and Performance Measurement in Operations Management: Dealing with the Metrics Maze. *Journal of Operations Management, 22*(3), 209-218.

Miles, M. B., & Huberman, A. M. (1994). *Qualitative Data Analysis: an Expanded Sourcebook*. Thousands Oaks: Sage Publications.

Myers, M. D. (1997). Qualitative Research in Information Systems. *MIS Quarterly, 21*(2), 241-242.

Myers, M. D., & Newman, M. (2007). The Qualitative Interview in IS Research: Examining the Craft. *Information and Organization, 17*(1), 2-26.

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). New York: McGraw-Hill.

Nahapiet, J., & Ghoshal, S. (1998). Social Capital, Intellectual Capital, and the Organizational Advantage. *Academy of Management Review, 23*(2), 242-266.

Napier, N. P., Keil, M., & Tan, F. B. (2009). IT Project Managers' Construction of Successful Project Management Practice: A Repertory Grid Investigation. *Information Systems Journal, 19*(3), 255-282.

Narayanan, S., Balasubramanian, S., & Swaminathan, J. M. (2009). A Matter of Balance: Specialization, Task Variety, and Individual Learning in a Software Maintenance Environment. *Management Science, 55*(11), 1861-1876.

Nelson, K. M., & Cooprider, J. G. (1996). The Contribution of Shared Knowledge to IS Group Performance. *MIS Quarterly, 20*(4), 409-429.

Nelson, K. M., Nadkarni, S., Narayanan, V. K., & Ghods, M. (2000). Understanding Software Operations Support Expertise: A Revealed Causal Mapping Approach. *MIS Quarterly, 24*(3), 475-507.

Nerur, S., Mahapatra, R., & Mangalaraj, G. (2005). Challenges of Migrating to Agile Methodologies. *Communications of the ACM, 48*(5), 72-78.

Northcraft, G. B., & Chase, R. B. (1985). Managing Service Demand at the Point of Delivery. *Academy of Management Review, 10*(1), 66-75.

Noume, S., Loic, F., & Setia, P. (2018). *IT Infrastructure Provisioning and IT Infrastructure Governance*. Paper presented at the International Conference on Information Systems, San Francisco, USA.

O'Reilly, C. A., & Tushman, M. L. (2008). Ambidexterity as a Dynamic Capability: Resolving the Innovator's Dilemma. *Research in Organizational Behavior, 28*, 185-206.

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.

Orlikowski, W. J. (1993). CASE Tools as Organizational Change: Investigating Incremental and Radical Changes in Systems Development. *MIS Quarterly*, 309-340.

Orlikowski, W. J., & Baroudi, J. J. (1991). Studying Information Technology in Organizations: Research Approaches and Assumptions. *Information Systems Research, 2*(1), 1-28.

Orlikowski, W. J., & Daniel, R. (1991). Information Technology and the Structuring of Organizations. *Information Systems Research, 2*(2), 143-169. doi:doi:10.1287/isre.2.2.143

Ouchi, W. G. (1978). The Transmission of Control through Organizational Hierarchy. *Academy of Management Journal, 21*(2), 173-192.

Ouchi, W. G. (1979). A Conceptual Framework for the Design of Organizational Control Mechanisms. *Management Science, 25*(9), 833-848.

Overby, E., Bharadwaj, A., & Sambamurthy, V. (2006). Enterprise Agility and the Enabling Role of Information Technology. *European Journal of Information Systems, 15*(2), 120-131.

Overhage, S., & Schlauderer, S. (2012). *How Sustainable are Agile Methodologies? Acceptance Factors and Developer Perceptions in Scrum Projects*. Paper presented at the European Conference on Information Systems, Barcelona, Spain.

Ozcan, P., & Eisenhardt, K. M. (2009). Origin of Alliance Portfolios: Entrepreneurs, Network Strategies, and Firm Performance. *Academy of Management Journal, 52*(2), 246-279.

Ozer, M., & Vogel, D. (2015). Contextualized Relationship Between Knowledge Sharing and Performance in Software Development. *Journal of Management Information Systems, 32*(2), 134-161.

Paddock, C. E. (1985). An Assessment of Productivity and Operations Control as Motives for Office Automation. *Journal of Management Information Systems, 1*(4), 76-86.

Patton, M. Q. (1990). *Qualitative Evaluation and Research Methods* (2nd ed.). Newbury Park, CA.: SAGE Publications

Peppard, J. (2010). Unlocking the Performance of the Chief Information Officer (CIO). *California Management Review, 52*(4), 73-99.

Peppard, J. (2018). Rethinking the Concept of the IS Organization. *Information Systems Journal, 28*(1), 76-103.

Peppard, J., & Ward, J. (2004). Beyond Strategic Information Systems: Towards an IS Capability. *The Journal of Strategic Information Systems, 13*(2), 167-194.

Persson, J. S., Mathiassen, L., & Aaen, I. (2012). Agile Distributed Software Development: Enacting Control Through Media and Context. *Information Systems Journal, 22*(6), 411-433.

Pflügler, C., Wiesche, M., & Krcmar, H. (2018). *Subgroups in Agile and Traditional IT Project Teams.* Paper presented at the Hawaii International Conference on System Sciences, Waikoloa Village, USA.

Powell, T. C. (1992). Organizational Alignment as Competitive Advantage. *Strategic Management Journal, 13*(2), 119-134.

Pries-Heje, L., & Pries-Heje, J. (2011). *Agile & Distributed Project Management: A Case Study Revealing why Scrum is Useful.* Paper presented at the European Conference on Information Systems, Helsinki, Finland.

Przybilla, L., Wiesche, M., & Krcmar, H. (2018). *The Influence of Agile Practices on Performance in Software Engineering Teams: A Subgroup Perspective*. Paper presented at the ACM SIGMIS-CPR, Buffalo, USA.

Rai, A., & Sambamurthy, V. (2006). Editorial Notes—The Growth of Interest in Services Management: Opportunities for Information Systems Scholars. *Information Systems Research, 17*(4).

Ramesh, B., Cao, L., & Baskerville, R. (2010). Agile Requirements Engineering Practices and Challenges: An Empirical Study. *Information Systems Journal, 20*(5), 449-480.

Ramesh, B., Mohan, K., & Cao, L. (2012). Ambidexterity in Agile Distributed Development: an Empirical Investigation. *Information Systems Research, 23*(2), 323-339.

Rdiouat, Y., Bahsani, S., Lakhdissi, M., & Semma, A. (2015). Measuring and Improving Information Systems Agility Through the Balanced Scorecard Approach. *International Journal of Computer Science Issues (IJCSI), 12*(5), 58.

Recker, J., Rosemann, M., Indulska, M., & Green, P. (2009). Business Process Modeling-A Comparative Analysis. *Journal of the Association for Information Systems, 10*(4), 333-363.

Reed, J. P. (2014). *DevOps in Practice*. Sebastopol, USA O'Reilly Media, Inc.

Reich, B. H., & Benbasat, I. (2000). Factors that Influence the Social Dimension of Alignment Between Business and Information Technology Objectives. *MIS Quarterly, 24*(1), 81-113.

Reynolds, P., & Yetton, P. (2015). Aligning Business and IT Strategies in Multi-Business Organizations. *Journal of Information Technology, 30*(2), 101-118.

Riemer, K., & Klein, S. (2008). Is the V-form the Next Generation Organisation? An Analysis of Challenges, Pitfalls and Remedies of ICT-Enabled Virtual Organisations Based on Social Capital Theory. *Journal of Information Technology, 23*(3), 147-162.

Ries, E. (2011). *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses*. New York Crown Business.

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.

Rivera, J., & van der Meulen, R. (2015). Gartner Says By 2016, DevOps Will Evolve From a Niche to a Mainstream Strategy Employed by 25 Percent of Global 2000 Organizations [Press release]. Retrieved from https://www.gartner.com/newsroom/id/2999017

Roche, J. (2013). Adopting DevOps Practices in Quality Assurance. *Communications of the ACM, 56*(11), 38-43.

Ross, J. W., Beath, C. M., & Goodhue, D. L. (1996). Develop Long-Term Competitiveness Through IT Assets. *MIT Sloan Management Review, 38*(1), 31.

Ross, J. W., Sebastian, I., Beath, C., Mocker, M., Moloney, K., & Fonstad, N. (2016). *Designing and Executing Digital Strategies*. Paper presented at the International Conference on Information Systems, Dublin, Ireland.

Rouse, M. (2016). DevOps. In *TechTarget*. Retrieved from http://searchitoperations.techtarget.com/definition/DevOps.

Runyan, R. C., Huddleston, P., & Swinney, J. L. (2007). A Resource-Based View of the Small Firm: Using a Qualitative Approach to Uncover Small Firm Resources. *Qualitative Market Research: An International Journal, 10*(4), 390-402.

Sabherwal, R., & Chan, Y. E. (2001). Alignment Between Business and IS Strategies: A Study of Prospectors, Analyzers, and Defenders. *Information Systems Research, 12*(1), 11-33.

Sabherwal, R., Hirschheim, R., & Goles, T. (2001). The Dynamics of Alignment: Insights from a Punctuated Equilibrium Model. *Organization Science, 12*(2), 179-197.

Salmela, H., Tapanainen, T., Baiyere, A., Hallanoro, M., & Galliers, R. (2015). *IS Agility Research: An Assessment and Future Directions*. Paper presented at the European Conference on Information Systems, Münster, Germany.

Sambamurthy, V., Bharadwaj, A., & Grover, V. (2003). Shaping Agility through Digital Options: Reconceptualizing the Role of Information Technology in Contemporary Firms. *MIS Quarterly, 27*(2), 237-263.

Sambamurthy, V., & Zmud, R. W. (1999). Arrangements for Information Technology Governance: A Theory of Multiple Contingencies. *MIS Quarterly, 23*(2), 261-290.

Sarker, S., Munson, C. L., Sarker, S., & Chakraborty, S. (2009). Assessing the Relative Contribution of the Facets of Agility to Distributed Systems Development Success: an Analytic Hierarchy Process Approach. *European Journal of Information Systems, 18*(4), 285-299.

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.

Sarker, S., Xiao, X., & Beaulieu, T. (2013). Guest Editorial: Qualitative Studies in Information Systems: A Critical Review and some Guiding Principles. *MIS Quarterly, 37*(4), iii-xviii.

Scheerer, A., & Kude, T. (2014). *Exploring Coordination in Large-Scale Agile Software Development: A Multiteam Systems Perspective*. Paper presented at the International Conference on Information Systems, Auckland, New Zealand.

Schermann, M., Wiesche, M., & Krcmar, H. (2012). The Role of Information Systems in Supporting Exploitative and Exploratory Management Control Activities. *Journal of Management Accounting Research, 24*(1), 31-59.

Schlauderer, S., Overhage, S., & Fehrenbach, B. (2015). *Widely Used but also Highly Valued? Acceptance Factors and Their Perceptions in Water-Scrum-Fall Projects*. Paper presented at the International Conference on Information Systems, Fort Worth, USA.

Schmidt, C., Kude, T., Heinzl, A., & Mithas, S. (2014). *How Agile Practices Influence the Performance of Software Development Teams: The Role of Shared Mental Models and Backup*. Paper presented at the International Conference on Information Systems Auckland, New Zealand.

Schwartz, R. B., & Russo, M. C. (2004). How to Quickly Find Articles in the Top IS Journals. *Communications of the Association for Information Systems, 47*(2), 98-101.

Sebastian, I. M., Ross, J. W., Beath, C., Mocker, M., Moloney, K. G., & Fonstad, N. O. (2017). How Big Old Companies Navigate Digital Transformation. *MISQ Executive, 16*(3), 197-213.

Sen, S., Raghu, T. S., & Vinze, A. (2009). Demand Heterogeneity in IT Infrastructure Services: Modeling and Evaluation of a Dynamic Approach to Defining Service Levels. *Information Systems Research, 20*(2), 258-276.

Shaft, T. M., & Vessey, I. (1998). The Relevance of Application Domain Knowledge: Characterizing the Computer Program Comprehension Process. *Journal of Management Information Systems, 15*(1), 51-78.

Shaft, T. M., & Vessey, I. (2006). The Role of Cognitive Fit in the Relationship Between Software Comprehension and Modification. *MIS Quarterly*, *15*(1), 29-55.

Sharma, S., & Rai, A. (2015). Adopting IS Process Innovations in Organizations: The Role of IS Leaders' Individual Factors and Technology Perceptions in Decision Making. *European Journal of Information Systems, 24*(1), 23-37.

Shpilberg, D., Berez, S., Puryear, R., & Shah, S. (2007). Avoiding the Alignment Trap in IT. *MIT Sloan Management Review, 49*(1), 51-58.

Silverthorne, V. (2017). Business Process Management Aids Application Development. In V. Silverthorne (Ed.), *Application Development Handbook*: TechTarget.

Slaughter, S. A., Levine, L., Ramesh, B., Pries-Heje, J., & Baskerville, R. (2006). Aligning Software Processes with Strategy. *MIS Quarterly*, *30*(1), 891-918.

Sommerville, I. (2009). *Software Engineering* (Vol. 9). Boston, USA: Pearson Education, Inc.

Spohrer, K., Gholami, B., & Heinzl, A. (2012). *Team Learning in Information Systems Development-A Literature Review*. Paper presented at the European Conference on Information Systems, Barcelona, Spain.

Spohrer, K., Kude, T., Schmidt, C. T., & Heinzl, A. (2013). *Knowledge Creation In Information Systems Development Teams: The Role Of Pair Programming And Peer Code Review*. Paper presented at the European Conference on Information Systems, Utrecht, Netherlands.

Stachour, P., & Collier-Brown, D. (2009). You Don't Know Jack About Software Maintenance. *Communications of the ACM, 52*(11), 54-58.

Ståhl, D., & Bosch, J. (2014). Modeling Continuous Integration Practice Differences in Industry Software Development. *Journal of Systems and Software, 87*, 48-59.

Star, S. L., & Ruhleder, K. (1996). Steps Toward an Ecology of Infrastructure: Design and Access for Large Information Spaces. *Information Systems Research, 7*(1), 111-134.

Staw, B. M. (1991). Dressing Up Like an Organization: When Psychological Theories can Explain Organizational Action. *Journal of Management, 17*(4), 805-819.

Stein, M.-K., Galliers, R. D., & Markus, M. L. (2013). Towards an Understanding of Identity and Technology in the Workplace. *Journal of Information Technology, 28*(3), 167-182.

Strauss, A., & Corbin, J. (1990). *Basics of Qualitative Research* (Vol. 15): Newbury Park, CA: Sage.

Subramaniam, M., & Youndt, M. A. (2005). The Influence of Intellectual Capital on the Types of Innovative Capabilities. *Academy of Management Journal, 48*(3), 450-463.

Sundaramurthy, C., & Lewis, M. (2003). Control and Collaboration: Paradoxes of Governance. *Academy of Management Review, 28*(3), 397-415.

Swan, J., Newell, S., & Robertson, M. (1999). The Illusion of 'Best Practice' in Information Systems for Operations Management. *European Journal of Information Systems, 8*(4), 284-293.

Swanson, B., & Dans, E. (2000). System Life Expectancy and the Maintenance Effort: Exploring their Equilibration. *MIS Quarterly*, 277-297.

Swanson, B. E., & Beath, C. M. (1989). Reconstructing the Systems Development Organization. *MIS Quarterly, 13*(3), 293-307.

Tan, W. G., & Gable, G. G. (1997). Maintaining Centralised Application Systems: A Cross-Country, Cross-Sector, Cross-Platform Comparison. *European Journal of Information Systems, 6*(4), 193-207.

Tarafdar, M., & Gordon, S. R. (2007). Understanding the Influence of Information Systems Competencies on Process Innovation: A Resource-Based View. *The Journal of Strategic Information Systems, 16*(4), 353-392.

Tate, G., Verner, J., & Jeffery, R. (1992). CASE: A Testbed for Modeling, Measurement and Management. *Communications of the ACM, 35*(4), 65-72.

Taylor, A., & Greve, H. R. (2006). Superman or the Fantastic Four? Knowledge Combination and Experience in Innovative Teams. *Academy of Management Journal, 49*(4), 723-740.

Taylor, M., Moynihan, E., & Wood-Harper, T. (1997). Knowledge for Software Maintenance. *Journal of Information Technology, 12*(2), 155-166.

Thorenz , L. (2019). Die IT-Trends bis 2024 von IDC, KI, Edge Computing, DevOps, Cloud. Retrieved from https://www.cio.de/a/die-it-trends-bis-2024-von-idc,3592203

Tilson, D., Lyytinen, K., & Sørensen, C. (2010). Research Commentary—Digital Infrastructures: The Missing IS Research Agenda. *Information Systems Research, 21*(4), 748-759.

Tiwana, A. (2009). Governance-Knowledge Fit in Systems Development Projects. *Information Systems Research, 20*(2), 180-197.

Tiwana, A. (2010). Systems Development Ambidexterity: Explaining the Complementary and Substitutive roles of Formal and Informal Controls. *Journal of Management Information Systems, 27*(2), 87-126.

Tiwana, A. (2012). Novelty-Knowledge Alignment: A Theory of Design Convergence in Systems Development. *Journal of Management Information Systems, 29*(1), 15-52.

Tiwana, A. (2018). Platform Synergy: Architectural Origins and Competitive Consequences. *Information Systems Research, 29*(4), 829-848.

Tiwana, A., & Keil, M. (2007). Does Peripheral Knowledge Complement Control? An Empirical Test in Technology Outsourcing Alliances. *Strategic Management Journal, 28*(6), 623-634.

Tiwana, A., & Keil, M. (2009). Control in Internal and Outsourced Software Projects. *Journal of Management Information Systems, 26*(3), 9-44.

Tiwana, A., & Konsynski, B. (2010). Complementarities between Organizational IT Architecture and Governance Structure. *Information Systems Research, 21*(2), 288-304.

Tiwana, A., Konsynski, B., & Venkatraman, N. (2013). Information Technology and Organizational Governance: The IT Governance Cube. *Journal of Management Information Systems, 30*(3), 7-12.

Torkar, R., Minoves, P., & Garrigós, J. (2011). Adopting Free/Libre/Open Source Software Practices, Techniques and Methods for Industrial Use. *Journal of the Association for Information Systems, 12*(1), 88.

Towry, K. L. (2003). Control in a Teamwork Environment—The Impact of Social Ties on the Effectiveness of Mutual Monitoring Contracts. *The Accounting Review, 78*(4), 1069-1095.

Trigg, R. H., & Bødker, S. (1994). *From Implementation to Design: Tailoring and the Emergence of Systematization in CSCW*. Paper presented at the Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work, Chapel Hill,, USA.

Tripp, J. F., Riemenschneider, C., & Thatcher, J. B. (2016). Job Satisfaction in Agile Development Teams: Agile Development as Work Redesign. *Journal of the Association for Information Systems, 17*(4), 267 – 307.

Trusson, C. R., Doherty, N. F., & Hislop, D. (2014). Knowledge Sharing Using IT Service Management Tools: Conflicting Discourses and Incompatible Practices. *Information Systems Journal, 24*(4), 347-371.

Tushman, M. L., & O'Reilly, C. A. (1996). The Ambidextrous Organizations: Managing Evolutionary and Revolutionary Change. *California Management Review, 38*(4), 8-30.

Urquhart, C. (2012). *Grounded Theory for Qualitative Research: A Practical Guide*. London: SAGE Publication Inc.

Urquhart, C., Lehmann, H., & Myers, M. D. (2010). Putting the 'Theory' Back into Grounded Theory: Guidelines for Grounded Theory Studies in Information Systems. *Information Systems Journal, 20*(4), 357-381.

Venkatesh, V., Brown, S. A., & Bala, H. (2013). Bridging the Qualitative-Quantitative Divide: Guidelines for Conducting Mixed Methods Research in Information Systems. *MIS Quarterly, 37*(1), 21-54.

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.

Vial, G., & Rivard, S. (2015). *Understanding Agility in ISD Projects*. Paper presented at the International Conference on Information Systems, Fort Worth, USA.

Vidgen, R., & Wang, X. (2009). Coevolving Systems and the Organization of Agile Software Development. *Information Systems Research, 20*(3), 355-376.

Vinekar, V., Slinkman, C. W., & Nerur, S. (2006). Can Agile and Traditional Systems Development Approaches Coexist? An Ambidextrous View. *Information Systems Management, 23*(3), 31-42.

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.

Walsham, G. (1995). Interpretive Case Studies in IS Research: Nature and Method. *European Journal of Information Systems, 4*(2), 74-81.

Wang, X., Conboy, K., & Pikkarainen, M. (2012). Assimilation of Agile Practices in Use. *Information Systems Journal, 22*(6), 435-455.

Wang, Z., Wang, N., & Liang, H. (2014). Knowledge Sharing, intellectual Capital and Firm Performance. *Management Decision, 52*(2), 230-258.

Wareham, J., Fox, P. B., & Cano Giner, J. L. (2014). Technology Ecosystem Governance. *Organization Science, 25*(4), 1195-1215.

Watson-Manheim, M.-B., Chudoba, K. M., & Crowston, K. (2002). Discontinuities and Continuities: A New Way to Understand Virtual Work. *Information Technology & People, 15*(3), 191-209.

Watts, S., & Henderson, J. C. (2006). Innovative IT Climates: CIO Perspectives. *The Journal of Strategic Information Systems, 15*(2), 125-151.

Webster, J., & Watson, R. T. (2002). Analyzing the Past to Prepare for the Future: Writing a Literature Review. In *MIS Quarterly* (Vol. 26, pp. 13-23).

Wei, H.-L., Wang, E. T. G., & Ju, P.-H. (2005). Understanding Misalignment and Cascading Change of ERP Implementation: A Stage View of Process Analysis. *European Journal of Information Systems, 14*(4), 324-334.

Weill, P., & Ross, J. W. (2004). *IT Governance: How Top Performers Manage IT Decision Rights for Superior Results*. Boston, USA: Harvard Business Press.

Weiser, M. (1982). Programmers Use Slices when Debugging. *Communications of the ACM, 25*(7), 446-452.

Wernerfelt, B. (1984). A Resource-Based View of the Firm. *Strategic Management Journal, 5*(2), 171-180.

West, D., Grant, T., Gerush, M., & D'silva, D. (2010). Agile Development: Mainstream Adoption has Changed Agility. *Forrester Research, 2*(1).

Wiedemann, A. (2017). *A New Form of Collaboration in IT Teams - Exploring the DevOps Phenomenon*. Paper presented at the Pacific Asia Conference on Information Systems, Langkawi, Malaysia.

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.

Wiedemann, A., & Weeger, A. (2017). *Developing Intellectual Capital within IT Teams: A Literature Review*. Paper presented at the European Conference on Information Systems, Guimarães, Portugal.

Wiedemann, A., & Wiesche, M. (2018a). *Are You Ready For DevOps? Required Skill Set for DevOps Teams*. Paper presented at the European Conference on Information Systems, Portsmouth, UK.

Wiedemann, A., & Wiesche, M. (2018b). *How to Implement Clan Control in DevOps Teams*. Paper presented at the Americas Conference on Information Systems, New Orleans, USA.

Wiener, M., Mähring, M., Remus, U., & Saunders, C. S. (2016). Control Configuration and Control Enactment in Information Systems Projects: Review and Expanded Theoretical Framework. *MIS Quarterly, 40*(3), 741-774.

Wiesche, M., Jurisch, M. C., Yetton, P. W., & Krcmar, H. (2017). Grounded Theory Methodology in Information Systems Research. *MIS Quarterly, 41*(3), 685-701.

Wiesche, M., & Krcmar, H. (2014). *The Relationship of Personality Models and Development Tasks in Software Engineering.* Paper presented at the ACM SIGMIS-CPR, Singapore, Singapore.

Wolfswinkel, J., Furtmueller, E., & Wilderom, C. (2013). Using Grounded Theory as a Method for Rigorously Reviewing Literature. *European Journal of Information Systems, 22*(1), 45-55.

Xue, L., Ray, G., & Gu, B. (2011). Environmental Uncertainty and IT Infrastructure Governance: A Curvilinear Relationship. *Information Systems Research, 22*(2), 389-399.

Yang, H., Antunes, P., & Tate, M. (2016). *Towards a Unified Conceptualisation of IS Agility*. Paper presented at the IEEE 20th International Conference on Computer Supported Cooperative Work in Design (CSCWD), Nanchang, China

Yin, R. K. (2009). *Case Study Research: Design and Methods* (Vol. 4). Thousand Oaks, CA: SAGE.

Yin, R. K. (2014). *Case Study Research: Design and Methods* (Vol. 5). Thousand Oaks, CA: Sage Publications Inc.

Yin, R. K. (2018). *Case Study Research and Applications: Design and Methods* (Vol. 6). Los Angeles: SAGE Publication Inc.

Yoo, Y., Boland Jr, R. J., Lyytinen, K., & Majchrzak, A. (2012). Organizing for Innovation in the Digitized World. *Organization Science, 23*(5), 1398-1408.

Yoo, Y., Henfridsson, O., & Lyytinen, K. (2010). Research Commentary—The New Organizing Logic of Digital Innovation: An Agenda for Information Systems Research. *Information Systems Research, 21*(4), 724-735.

Appendix A – Software Operations Literature

No.	Paper
1	Alter (2008)
2	Alter (2013)
3	April et al. (2005)
4	Banker et al. (1991)
5	Banker et al. (1998)
6	Bardhan et al. (2010)
7	Benaroch and Chernobai (2017)
8	Boonstra et al. (2017)
9	Constantinides and Barrett (2014)
10	Cram and Brohman (2013)
11	Cram et al. (2016a)
12	Dekleva (1992)
13	Dennis et al. (2014)
14	Edberg et al. (2012)
15	Fielt et al. (2013)
16	Fladmoe-Lindquist and Jacque (1995)
17	Goldstein et al. (2011)
18	Henfridsson and Bygstad (2013)
19	Hipkin (1996)
20	Ives and Vitale (1988)
21	Jia and Reich (2013)
22	Kim and Westin (1988)
23	Koutsikouri et al. (2018)
24	Lempinen and Rajala (2014)
25	Markus and Keil (1994)
26	Melnyk et al. (2004)
27	Narayanan, Balasubramanian, and Swaminathan (2009)
28	Nelson et al. (2000)
29	Northcraft and Chase (1985)
30	Noume, Loic, and Setia (2018)
31	Paddock (1985)
32	Rai and Sambamurthy (2006)
33	Sen, Raghu, and Vinze (2009)
34	Shaft and Vessey (1998)
35	Shaft and Vessey (2006)
36	Sharma and Rai (2015)
37	Slaughter et al. (2006)
38	Stachour and Collier-Brown (2009)
39	Star and Ruhleder (1996)

40	Swan, Newell, and Robertson (1999)
41	Swanson and Beath (1989)
42	Swanson and Dans (2000)
43	Tan and Gable (1997)
44	Taylor, Moynihan, and Wood-Harper (1997)
45	Tilson et al. (2010)
46	Tiwana et al. (2013)
47	Trusson et al. (2014)
48	Xue, Ray, and Gu (2011)
49	Yoo et al. (2010)

Table 61. Software Operations and Maintenance Papers (Own Depiction)

Appendix B – All Publications

Publications: Scientific Journals (Peer Reviewed)

Wiedemann, A., Wiesche, M., Gewald, H., & Krcmar, H. (2020). Understanding How DevOps Aligns Development and Operations: A Tripartite Model of Intra-IT Alignment. **European Journal of Information Systems**, 1-16.

Wiedemann, A., Forsgren, N., Wiesche, M., Gewald, H., and Krcmar, H. (2019). "Research for Practice: The DevOps Phenomenon," **Communications of the ACM (**62 8), pp. 44-49.

Wiedemann, A., Forsgren, N., Wiesche, M., Gewald, H., and Krcmar, H. (2019). „The DevOps Phenomenon – An Executive Crash Course," **ACM Queue**, March/April.

Wiedemann, A., and Gewald, H. 2017. "Examining Cross-Domain Alignment: The Correlation of Business Strategy, IT Management, and IT Business Value," **International Journal of IT/Business Alignment and Governance** (8:1), pp. 17-31.

Wiedemann, A., Weeger, A., and Gewald, H. (2015). "Organizational Structure Vs. Capabilities: Examining Critical Success Factors for Managing IT Service Delivery " **International Journal of IT/Business Alignment and Governance** (6:1), pp. 49-69.

Publications: Scientific Conferences (Peer Reviewed)

Wiedemann, A., Wiesche, M., Thatcher, J. B., and Gewald, H. (2019). A Control-Alignment Model for Product Orientation in DevOps Teams–A Multinational Case Study. in: **International Conference on Information Systems** (ICIS), Munich, Germany.

Wiedemann, A., Wiesche, M., and Krcmar, H. (2019). „ Integrating Development and Operations in Cross-Functional Teams-Toward a DevOps Competency Model," in: **Computers and People Research Conference**, ACM, Nashville, USA.

Wiedemann, A., Wiesche, M., Gewald, H., and Krcmar, H. (2019). „Implementing the Planning Process within DevOps Teams to Achieve Continuous Innovation," in: **Hawaii International Conference on System Sciences** (HICSS), Grand Wailea, USA.

Wagner, H.-T., Gewald, H., Moos, B., and Wiedemann, A. (2019). "The Influence of Inter-Organizational Alignment on Consultancy Project Success," in: **Americas Conference on Information Systems** (AMCIS). Cancun, Mexicó.

Wiedemann, A., Wiesche, M., Gewald, and H., Krcmar, H. (2018). „Integrating DevOps within IT Organizations – Key Pattern of a Case Study", in: **GI-Fachtagung Projektmanagement und Vorgehensmodelle**, Düsseldorf, Germany.

Wiedemann, A., and Wiesche, M. (2018). "How to Implement Clan Control in Devops Teams," in: **Americas Conference on Information Systems** (AMCIS). New Orleans, USA.

Wiedemann, A., and Wiesche, M. (2018). „Are You Ready for DevOps? Required Skill Set for DevOps Teams," in: **European Conference on Information Systems** (ECIS), Portsmouth, UK.

Wiedemann, A. (2018). “IT Governance Mechanisms for DevOps Teams – How Incumbent Companies Achieve Competitive Advantages,” in: **Hawaii International Conference on Systems Sciences** (HICSS), Waikoloa Village, USA.

Wiedemann, A., and Schulz, T. (2017). "Key Capabilities of DevOps Teams and Their Influence on Software Process Innovation: A Resource-Based View," in: **Americas Conference on Information Systems** (AMCIS), Boston, USA.

Wiedemann, A. (2017). “A New Form of Collaboration in IT Teams – Exploring the DevOps Phenomenon,” in: **Pacific Asia Conference on Information Systems** (PACIS), Langkawi, Malaysia.

Wiedemann, A., and Weeger, A. (2017). "Developing Intellectual Capital within Agile IT Teams: A Literature Review," in: **European Conference on Information Systems** (ECIS), Guimarães, Portugal.

Wiedemann, A., Gewald, H., and Weeger, A. (2017). "How IT Management Profile and IT Business Value Correlate – Exploring Cross-Domain Alignment," in: **Hawaii International Conference on System Sciences** (HICSS), Waikoloa Village, USA.

Wiedemann, A., and Weeger, A. (2016). "How to Design an IT Department? A Review and Synthesis of Key Characteristics," in: **Americas Conference on Information Systems** (AMCIS), San Diego, USA.

Wiedemann, A., Weeger, A., and Gewald, H. (2015). "Organizational Structure Vs. Capabilities: Examining Critical Success Factors for Managing IT Service Delivery," in: **Hawaii International Conference on System Sciences** (HICSS), Kauai, USA.

Appendix C – Published Articles in Original Format (P1-P8)

DEVELOPING INTELLECTUAL CAPITAL WITHIN AGILE IT TEAMS: A LITERATURE REVIEW

Research paper

Anna Wiedemann, Neu-Ulm University of Applied Sciences, Neu-Ulm, Germany, anna.wiedemann@hs-neu-ulm.de

Andy Weeger, Neu-Ulm University of Applied Sciences, Neu-Ulm, Germany, andy.weeger@hs-neu-ulm.de

Abstract

Since the Agile Manifesto was presented in 2001, more and more organizations move from traditional, plan-driven software development to agile approaches. This movement is motivated by the fact that environments are changing quickly and new requirements need fast implementation. We conducted a structured literature review to identify the current state of knowledge about agile IT teams and how they develop ambidextrous organizational learning to respond to rapid changes.

We draw on the intellectual capital theory with the aim to explore key capabilities of agile IT teams of prior research. Afterwards, we synthesize the key capabilities considering intellectual capital. We derive intellectual capital configurations that enable IT teams to develop ambidextrous organizational learning. Furthermore, we identified technological oriented capabilities of infrastructure flexibility and architecture modularity for agile IT teams. Therefore, we built the concept of technological capital and arranged these capabilities. Thus, this study contributes to research by highlighting the characteristics that enable IT teams to be agile and thus helping companies to gain competitive advantage. Furthermore, we discuss possibilities how balance in ambidextrous organizational learning could be achieved. Additionally, we provide further research opportunities in this research stream.

Keywords: Agile IT teams, ambidextrous organizational learning, intellectual capital, literature review

1 Introduction

Following the publication of the Agile Manifesto in 2001, Agile Software Development (ASD) methodologies became very popular (Vial and Rivard, 2015) as they provide an effective method to react to rapidly changing customer requirements in challenging environments (Hummel et al., 2013b). Hence, more and more organizations decided to move from plan-driven software development to agile approaches (Dybå and Dingsøyr, 2008; West et al., 2010; Tripp et al., 2016). Agility is defined as "*the ability to respond operationally and strategically to changes in the external environment"* (Fink and Neumann, 2007, p. 444). One of the most important differences between organizations that follow agile approaches and organizations that follow more traditional approaches is that they establish autonomous, self-organized teams, facilitate learning beyond knowledge silos and thus, facilitate and advance decisions-making within these teams (Coyle et al., 2015). For example, Amazon and Spotify replaced their traditional "silo" functions, which required enormous coordination efforts, with autonomous, cross-functional, product-centric teams that include a maximum of eight people (Advance IT Minnesota, 2016). This enables them to gain, share and implement knowledge, speed up decision making processes significantly and thus, to meet demand in rapid changing environments (Ramesh et al., 2012).

Developing innovative capabilities and realizing agile IT structures frequently enables companies to gain competitive advantages (Ramesh et al., 2012). However, companies need to be able to implement agile approaches to benefit from higher agility as reflected by shorter development cycles and strong collaboration within the IT team (Kude et al., 2014). This seems to be difficult for rather old-fashioned companies. In contrast to modern organizations like Amazon and Spotify, these companies are more likely used to long software delivery lifecycles resulting from strict procedures and the realization of predefined plans (Fitzgerald, 1998; Sommerville, 2009; Kude et al., 2014; Coyle et al., 2015). In other words, these companies are quite effective in utilizing plans to refine and exploit existing knowledge, but have difficulties to bridge knowledge boundaries.

Prior research noted that organizational learning –the process of adoption and integration of new knowledge– is critical for realizing change (Kang and Snell, 2009). Since the publication of March (1991) a lot of investigations on organizational learning concentrate on the approaches of exploration and exploitation (Kang et al., 2007; Kang and Snell, 2009). Exploration is defined as the learning of new knowledge outside from companies knowledge stocks, whereas exploitation focus on improving and deepening existing companies knowledge domains (March, 1991). But firms are struggling to practice exploration and exploitation together. Hence, literature emphasizes the concept of ambidexterity within organizations to balance both, exploration and exploitation (Gupta et al., 2006) to foster organizational learning (March, 1991). Furthermore, ambidextrous learning is contingent on an organization's intellectual capital (IC). The several dimension of IC, human, social, and organizational capital play a significant role within the learning process (Kang and Snell, 2009).

In order to develop the ability to respond to changes in the external environment (i.e., the ability to be agile), IT teams need to perform ambidextrous learning that is balancing exploration of new, as well as exploitation of existing knowledge and related resources (Lee et al., 2015). Although there is much literature on agile development teams, IS research is hitherto not clear on how IT teams could be enabled to develop this ability. To contribute to this gap, this paper sets out to synthesize prior research to illuminate how IT teams can be enabled to develop the ability of ambidextrous learning regarding the dimensions of IC. The following research questions guide this effort: *How can IT teams develop ambidextrous organizational learning capability to respond efficiently to environmental changes?*

We conducted a structured literature review to summarize the current state of knowledge and to approach this question. We analyze the findings of prior literature considering intellectual capital theory. More precisely, we analyze characteristics of agile teams that prior research has identified. Ultimately, we derive IC configurations that enable IT teams to develop ambidextrous organizational learning based on the approach of Kang and Snell (2009). Thus, this paper contributes to prior research by highlighting the characteristics that enable IT teams to be agile and thus, helping companies to gain competitive advantage.

The remainder of this paper is structured as follows. First, we discuss our theoretical understanding of agile IT teams, intellectual capital theory and ambidexterity in organizational learning. Then we outline our methods, present our findings and discuss the theoretical and practical implications of this research.

2 Theoretical Background

2.1 Agile IT Teams

Software development methodology is defined as an advocated approach for phases, rules, techniques, processes, documentation and training, employed for developing a software (Avison and Fitzgerald, 2003). In traditional development approaches, a sequential progression of phases is used and tasks are given to individual persons within separated organizational entities (e.g., business analyst unit, developer unit). Furthermore, this approach requires a huge amount of documentation, and much communication between the project participants is formalized through that documentation (Nerur et al., 2005).

Many of rather traditional structured IT organizations move to an agile, product-oriented structure of their entities to enable rapid software development. Normally, organizational entities are structured along several agile teams. An agile IT team is defined as a group of colleagues that work together with the aim to alter a running software or to build a new information system. Agile IT teams usually utilize agile methodologies such as Kanban, or Scrum, which create new possibilities to react fast with short development cycles and deliver software features iteratively to change customer or market requirements (Maruping et al., 2009).

Within agile IT teams all members should be able to perform all activities that are required to achieve the team objectives. Thus, members of agile teams should develop and advance a broad set of capabilities such as development, architecture knowledge, project management capabilities, and system administration (Tripp et al., 2016). Moreover, teams need to engage in continuous organizational learning processes to increase their performance. On the one hand, they are in the need to continually exploit existing knowledge, skills and capabilities (i.e., existing intellectual capital) and, on the other hand, to explore new knowledge such as that of their colleagues and customers (i.e., enhanced intellectual capital). Prior research shows that teams, which implement learning processes on group level early on, outperform other teams concerning both effectiveness and efficiency (Spohrer et al., 2012). Below we discuss the concepts of organizational learning and intellectual capital.

2.2 Intellectual Capital and Organizational Learning

Organizational learning consists of gaining, sharing, and integrating knowledge from outside the firm, as well as internal knowledge to improve on *"actions through better knowledge and understanding"* (Fiol and Lyles, 1985, p. 803). March (1991) provided a seminal of organizational learning that presents the trade-offs between exploration and exploitation, this model is widely used by existing literature (e.g. Gupta et al. (2006), Crossan et al. (1999)). Organizational learning appears on individual, group, and organizational level (Crossan et al., 1999; Kang and Snell, 2009) and involves the ability for exploitation and exploration. Exploitation focuses on a close, local, and deep search with repeating mechanisms to identify adequate solutions appropriate to firm's existing knowledge. Exploration is the result of a wide and generalized search to extend the knowledge fields of an organization into novel domains (Kang and Snell, 2009). The concept of ambidexterity refers to the ability to balance both exploration and exploitation. Consequently, literature on organizational ambidexterity focuses how these can be coordinated (Gupta et al., 2006).

As discussed above, agile, cross-functional IT teams are in the need for both exploration and exploitation within their teams (Ramesh et al., 2012). They need to exploit their skills and knowledge to develop the system in time and on budget. Organizations are able to encourage learning within subunits, since this could lead to the exploration of new possibilities (Fang et al., 2010). Moreover, they need to sense the environment, acquire new knowledge and skills to sense or anticipate, understand and respond to emerging changes (exploration) (Cao et al., 2009). Firms need an ambidextrous learning structure whereas they explore opportunities for new service development, as well as efficiently exploit existing products and services (O'Reilly and Tushman, 2008; Fang et al., 2010).

Prior research shows that organizational entities' abilities to develop or enhance organizational learning is closely tied to their knowledge inventory. In this regard, IC reflects this knowledge inventory, involving knowledge resources and capabilities of the networks the organization is embedded in. Consequently, IC refers to the combined knowledge that organizations can draw on to create a competitive advantage (Nahapiet and Ghoshal, 1998; Subramaniam and Youndt, 2005; Wang et al., 2014). Concerning IT teams, it can be assumed that their ability to develop ambidextrous learning and thus to respond to changes in the external environment (i.e., the ability to be agile) is contingent on their IC.

Prior research has contributed a broad range of conceptualizations of IC. The perhaps most-widely accepted framework conceptualizes IC as consisting of three dimensions: human capital, social capital, and organizational capital (Subramaniam and Youndt, 2005). These dimensions reflect knowledge domains which are aggregated and distributed among the organization, its individuals, and on relationships between individuals (Subramaniam and Youndt, 2005; Kang and Snell, 2009). We assume that there are IC configurations that contribute to the ability of IT teams to develop both forms of organizational learning, exploration as well as exploitation and thus enable them to be agile. Kang and Snell (2009) provided a multilevel model of the dimensions of IC and discussed how they can be combined to foster ambidextrous organizational learning. Their research based on March (1991), they present two alternatives for each dimension of IC. Below we discuss the dimensions of IC and their alternatives.

Human capital refers to the combined knowledge of the individuals within a firm. This dimension encompasses the knowledge, capabilities as well as skills, and abilities which belong to and are used by the individuals (Subramaniam and Youndt, 2005). Concerning organizational learning, human capital can be either specialist and generalist in nature. Specialists possess deep, bounded, and embedded knowledge in special domains, whereas generalists are multi-skilled, can draw on a manifold repertoire of competencies and are able to apply these in different knowledge domains (Taylor and Greve, 2006; Kang and Snell, 2009). Specialized human capital may be accompanied by a functional bias which can decrease individual readiness and ability to vary and change new knowledge within their specified domain (Dougherty, 1992). Hence, specialists might rather focus on exploitation and ignore exploration (Kang and Snell, 2009). Generalists are more broadly oriented in various knowledge domains, generalist human capital has less functional bias. Furthermore, they can provide multiple solutions for a problem and apply, discover, and combine new knowledge. Subsequently, generalist are rather likely to focus on exploratory learning (Taylor and Greve, 2006; Kang and Snell, 2009).

Social capital reflects the knowledge that is available via relational networks. It is the pipeline for knowledge exchange and combination within the company. Firms can gain advantages through sharing knowledge and insights and/or mental models within working groups and social relationships (Nahapiet and Ghoshal, 1998; Kang and Snell, 2009; Karahanna and Preston, 2013). In this regard, Ghobadi and Mathiassen (2016) show that shared knowledge and communication within agile IT teams can positively influence project success, failure rates, and risk factors. Social capital comprises three dimensions; structural, relational, and cognitive (Nahapiet and Ghoshal, 1998; Wagner et al., 2014). The structural dimension encompasses the overall scheme of network connections between individuals. This includes the communication settings between IT employees and other employees in form of meetings etc. (Wagner et al., 2014). The relational dimension of social capital contains the interpersonal exchange among individuals e.g. trust. This dimension focusses on the relationships between the individuals, which is essential for knowledge exchange and combination. The cognitive dimension implies to the degree of shared representations, understanding of individuals, and shared knowledge among individuals. Structural linking enhances relational and cognitive linkage and complete each other and social capital delivers the possibility for knowledge exchange (Nahapiet and Ghoshal, 1998; Kang and Snell, 2009; Karahanna and Preston, 2013). Prior research identified two generic social capital configurations: the cooperative and the entrepreneurial relational archetype of social capital. The cooperative relational archetype may facilitate exploitations as it involves strong and closely connected social networks characterized by strong trust and shared understanding (Kang et al., 2007). In contrast, the entrepreneurial relational archetype refers to more loosely coupled social networks. It enhances flexibility that is required to extent, gain, and absorb new knowledge and hence, it is expected to facilitate organizational exploratory learning.

Lastly, organizational capital refers to the institutional knowledge that is embedded in processes, structures, and systems. Organizational capital captures and integrates individual knowledge into organizational knowledge to be less dependent on individual employees (Grant, 1996; Kang and Snell, 2009). Based on prior literature, organizational capital can be categorized into two types: mechanistic and organic organizational capital (Kang and Snell, 2009). Mechanistic organizational capital refers to standardized structures and processes as well as tight rules. These are aimed at enhancing collaboration by determining enrooted patterns of interdependencies and providing detailed routines. Rules and standards facilitate integration of common opinions among colleagues. Thus, mechanistic organizational capital expected to facilitate exploitative organizational learning. Contrary, organic organizational capital refers to rather informal evolving rules, processes, and expectations about work results.

Organic organizational capital enables continuous exploration, absorption and integration of novel knowledge (Kang and Snell, 2009) and is hence, related to explorative organizational learning.

Overall, prior research indicates that the ability to respond efficiently to environmental changes (i.e., being agile) requires ambidextrous organizational learning. Thus, we assume that characteristics of agile teams contribute to the ability of ambidextrous organizational learning. Ambidextrous organizational learning, in turn, seems to be contingent on the IC of an organization or organizational entity like an IT team respectively. Hence, we assume that there are IC configurations that facilitate ambidextrous organizational learning and help IT teams being agile.

3 Method

In general, literature reviews provide overviews on current states of research in distinct fields, synthesize existing knowledge and identify research gaps and unexplored research questions. As to that, reviewing literature is a necessary process in scholarship, which ensures that new research is connected to the existing body of knowledge within a research topic (Webster and Watson, 2002; Wolfswinkel et al., 2013).

Until now, no research was identified that synthesizes the characteristics that enable agile IT teams to develop ambidextrous organizational learning capability that are required to respond efficiently to environmental changes. Thus, we conducted a systematic literature review aimed at synthesizing the characteristics identified by prior research, to identify avenues for further research and to provide a framework that could guide such research.

The guidelines posed by Webster and Watson (2002) guided our efforts. Following their advice, we first defined criteria for inclusion or exclusion of articles. Our review explicitly focuses on characteristics that enable IT teams to develop ambidextrous organizational learning and hence contribute to their ability being agile. More precisely, we aim to come up with a classification of intellectual capital that enables agility. Hence, the emphasis of this review is on past investigations of the design of agile organizational entities related to IT (e.g., IT teams and IT departments). Particularly, we were looking for articles describing characteristics of intellectual capital of IT related organizational entities that enable ambidextrous organizational learning. Table 1 presents our criteria for excluding papers.

Reason for exclusion
Research with strong focus of traditional IT teams/functions
Research that give no insights into agile information systems development
Research with strong focus of organizational agility without delivering any insights into agile IT teams
Research which deliver no attributes/capabilities/components or insights for the design of agile IT

Table 1. Exclusion criteria.

Webster and Watson (2002) recommend to start the literature search with important journals and conference proceedings and to conduct a forward and backward search of selected articles. Since keywords like “agile”, “agility” and “IT or IS teams” resulted in too many irrelevant articles, which, in turn, increases the chance of missing articles that prove value to our research questions, we decided not to use a key-word based search strategy. Rather, we decided to manually scan the table of contents of premier IS research journals from North America and Europe (Schwartz and Russo, 2004; Wolfswinkel et al., 2013). Following the advice of Webster and Watson (2002), we identified relevant articles by scanning the title and abstract of each article within the selected journals and conference proceedings. If heading, key words and abstract suited our criteria, papers were included for further review. Following this strategy, 93 articles have been identified.

We included eight journals recommended by the Association for Information Systems (AIS) senior scholars group in 2011 (i.e., the senior scholars basket of eight) plus MISQ Executive. We included the latter one because it contains frequently cited practice based publications in IS and is well respected in practice and academia. Furthermore, we included the past five years of the conference proceedings of the International Conference on Information Systems (ICIS) and the European Conference on Information Systems in our analysis (ECIS). By including the A- or B-rated conference proceedings, we intend to ensure that our literature review captures the latest research, which is not yet published in journals. We started our search in 2001, the year the Agile Manifesto was presented. We scanned all identified, published articles (93 articles) as described above and applied our exclusion criteria. Overall, 45 papers have been excluded. For the remaining 48 articles, we performed a backward research by checking the articles cited within the paper. Moreover, we used Web of Science and Google Scholar to perform a forward search to identify articles in other journals that cited these articles. These results have been analyzed using the same inclusion and exclusion criteria resulting in another 16 articles included in our analysis. In summary, we selected 64 papers (see Table 2).

Journal	**Coverage**	**Hits**	**Included**
MIS Quarterly	2001-2016	5	3
Journal of Strategic Information Systems	2001-2016	6	2
Journal of Management Information Systems	2001-2016	7	2
Journal of Information Technology	2001-2016	3	0
European Journal of Information Systems	2001-2016	11	6
Information Systems Journal	2001-2016	6	4
Information Systems Research	2001-2016	12	8
Journal of the AIS	2001-2016	9	5
MIS Quarterly Executive	2001-2016	5	3
International Conference on Information Systems	2011-2016	13	8
European Conference on Information Systems	2011-2016	16	7
Others Sources (forward/backward search)			16
Total		**93**	**64**

Table 2. Considered sources and number of identified articles.

We coded the selected papers following the guidelines of Webster and Watson (2002). To mitigate the risk of coding bias, the research team has developed rigor rules that guided the coding and analysis process. One researcher went through the paper and coded passages covering information about the focus and theoretical base of the paper, considered characteristics of agile IT teams and the primary findings. During analysis, regular meetings were set up to discuss emergent findings, issues and divergent views, until agreement regarding the constructs reflecting intellectual capital dimensions has been achieved.

4 Findings

Several researchers investigated different settings of agile IT departments or agile IT teams (Fink and Neumann, 2007; Baumgart et al., 2015; Vial and Rivard, 2015; Tripp et al., 2016). Analyzing and comparing the theoretical and empirical findings of these papers, we could identify eleven distinct characteristics related to intellectual capital, which help to foster ambidextrous organizational learning and thus, agility within organizational IT entities. The three dimensions of IC provide means for structuring the findings. Overall, literature confirms our assumption that IT teams need to employ both forms of organizational learning in order to be agile. Furthermore, we identified technology-oriented capabilities that could not be structured in one of the dimension of IC. Consequently, we added a technological capital dimensions. However, one could argue that technology could be closely related to the organizational dimension, we find much value in separating technological capital. Particularly since we are concerned with the IC of IT-related organizational entities.

Several papers within **human capital** related characteristics of agile IT teams like diversity, attitude, and requirements understanding. For example Lee and Xia (2010) highlighted, that agile IT teams that have a great diversity are more effective in responding to rapid environmental chances than homogenous teams. Literature emphasizes that the agile software teams' variety should fit the environment in which it is embedded, because diversity enhances the team's ability to react to changing environments.

Social capital encompasses the constructs shared knowledge, communication, as well as trust. Pries-Heje and Pries-Heje (2011) found that agile IT teams use effective communication ways and create trust. They conclude that this fosters social team capital. Social capital enhances IT teams understanding of business. Hence, this fosters knowledge exchange, enhances collaboration and accelerates strong ties within a network. Additionally, the team can achieve a better understanding of customer requirements through shared knowledge, trust, common language and communication (Wagner et al., 2014).

We identified governance, environment dynamism, and agile methodology use as constructs for agile IT teams that reflect characteristics of **organizational capital**. For example, Tiwana and Konsynski (2010) argue that enhancing agility is possible through IT governance decentralization. IT performance can increase through easing the exchange within an IT department (e.g. through decentralization).

Concerning **technological capital**, we found that prior research particularly investigates how infrastructure flexibility and architecture modularity enables agility. For example, new trends in technology faces infrastructure with great challenges, hence, a high degree of flexibility plays an essential role within agile IT teams. Prior studies associated IT infrastructure flexibility with fast business changes (Broadbent et al., 1999; Salmela et al., 2015), and enhanced organizational responsiveness (Bhatt et al., 2010; Salmela et al., 2015). These advantages are delivered by agile IT teams.

The following Table 3 presents the findings from the literature review structured per IC.

Human Capital		
Characteristic	**Definition of Construct**	**References**
Diversity	Is defined as the extent to which the team members are different to each other, especially with focus on their skills, experience, and functional background that are necessary to solve problems, thinking in multiple ways and implementing solutions. Team diversity enhances the use of different talents.	(Lee and Xia, 2010; Spohrer et al., 2012; Kude et al., 2014; Tripp et al., 2016)
Attitude	Is defined as the perception, feelings, behaviors, and commitment of individuals towards the agile approach within their IT team.	(Cao et al., 2009; Spohrer et al., 2013; Schmidt et al., 2014; Ghobadi and Mathiassen, 2016)
Requirements understanding	Is defined as the understanding of an agile IT team of customer's affordance for new features or necessary changes to enrich the software. Requirements have to be handled and implemented by the team and are likely to change during the development within an agile approach. A sequential collecting and refinement of requirements is necessary during the whole development.	(Fruhling and Vreede, 2006; Vidgen and Wang, 2009; Torkar et al., 2011; Overhage and Schlauderer, 2012; Cao et al., 2013; Matook and Maruping, 2014)

Social Capital		
Characteristic	**Definition of Construct**	**References**
Shared knowledge	Is defined as the team members shared, organized understanding of knowledge of essential key elements, roles and responsibilities, as well as tasks and skills of the environment of the team.	(Sambamurthy et al., 2003; Schmidt et al., 2014; Ghobadi and Mathiassen, 2016)
Communication	Is defined as the effective way to convey information between the team members. A clear communication and coordination process is dependent from the agile methodology use and is essential to avoid conflicts and redundant work. Agile team communication increases team members exchange frequency through short daily meeting, customer talks, and face-to-face communication.	(Hovorka and Larsen, 2006; Harris et al., 2009; Botzenhardt et al., 2011; Torkar et al., 2011; Li and Maedche, 2012; Wang et al., 2012; Hummel et al., 2013a; Scheerer and Kude, 2014; Schlauderer et al., 2015; Vial and Rivard, 2015; Tripp et al., 2016)
Trust	Is defined as the ability to show widespread respect and to have a trustfulness relationship between two individuals or within the agile IT team.	(Baskerville and Pries-Heje, 2004; Dybå and Dingsøyr, 2008; Goh et al., 2013; Matook and Maruping, 2014; Baumgart et al., 2015)
Organizational Capital		
Characteristic	**Definition of Construct**	**References**
Governance	Is defined as the degree of decentralization of decision-making authority within self-organized team-based structures of the IT department. Further, agile IT teams have a great autonomy in the decision making power.	(Tiwana and Konsynski, 2010; Balaji et al., 2011; Botzenhardt et al., 2011; Torkar et al., 2011; Coyle et al., 2015; Krancher and Luther, 2015)
Environment dynamism	Is defined as the agile IT team shared perceptions of practices, techniques, and procedures of an organization that are used to react on unpredictable and rapid changes of customers' preferences.	(Watts and Henderson, 2006; Sarker et al., 2009; Lee et al., 2015; Lowry and Wilson, 2016)
Agile methodology use	Is defined as the extent to which the IT team uses agile software development methods as well as agile project management practices. Agile methodologies facilitate the rapid application development; have as little as possible documentation, active involvement of customers, and interaction and feedback processes as well as a strong focus on short cycle releases.	(Baskerville and Pries-Heje, 2004; Fitzgerald et al., 2006; Fruhling and Vreede, 2006; Ågerfalk et al., 2009; Balijepally et al., 2009; Cao et al., 2009; Mangalaraj et al., 2009; Maruping et al., 2009; Wang et al., 2012; Tripp et al., 2016)
Technological Capital		
Characteristic	**Definition of Construct**	**References**

Infrastructure flexibility	Is defined as the ability of an IT team to react quickly and cost efficiently to customer demands. The infrastructure is flexible and compatible for emerging technologies and able to support running systems.	(Fink and Neumann, 2007; Bush et al., 2010; Goh et al., 2013; Rdiouat et al., 2015; Salmela et al., 2015)
Modular architecture	Is defined as the degree to which an IT team can work within an architecture that disaggregated into autonomous subsystems to foster communication through standardized interfaces. This encompasses the loose coupling of applications, which means that changes in one system does not affect another system as well as competencies in emerging technologies.	(Fink and Neumann, 2007; Choi et al., 2010; Tiwana and Konsynski, 2010; Yang et al., 2016)

Table 3. Findings of the literature review: Key constructs of agile IT teams.

The framework resulting from these findings is presented in Figure 1 below:

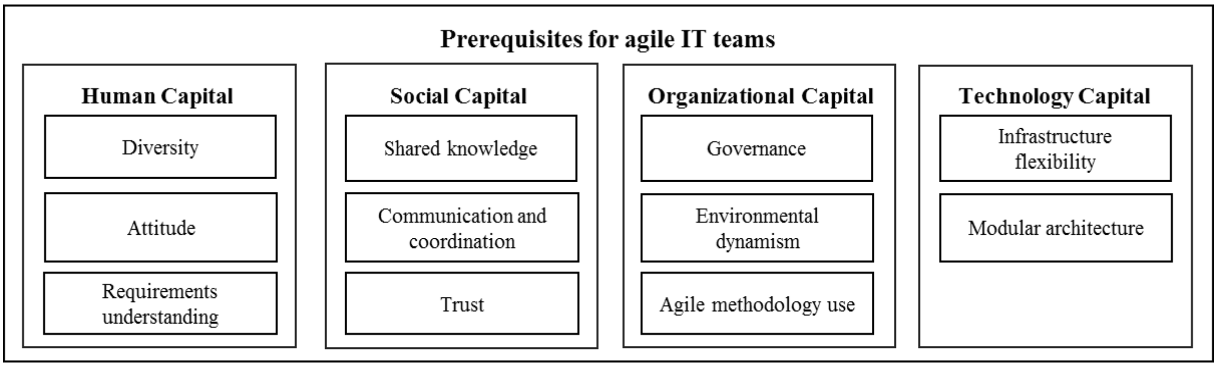


Figure 1. Findings: Four dimensions of key constructs.

The literature review largely confirmed our assumption. The results show the characteristics of agile IT teams contribute to ambidextrous organizational learning, which is dependent from IC. In the following, we present our findings for the different configurations of IC as well as for technological capital.

Generalist vs. specialist human capital: During the literature review, the research team identified different constructs, which reflect human capital characteristics. Individuals are important to uncover organizational boundaries and hence, foster the ability to receive and apply knowledge (Subramaniam and Youndt, 2005). Within agile IT teams, individual people with requisite special skills are one of the most important factors for a successful mode of operation. For example Chow and Cao (2008) explained that people based skills like high competency (diversity) are required to complete an agile project on time. This research shows that both aspects of human capital of organizational learning need to be given within agile IT teams. Furthermore, team members need an attitude, which is compliant with the mode of operation of the team. The team members need a commitment about the agile setting (Cao et al., 2009). Additionally, agile IT teams need general knowledge across a broad range of tasks that need to be delivered by the team (e.g. requirements understanding). IT teams should be able to represent all roles of their team that are necessary to complete their responsibilities (Tripp et al., 2016). This is a typical characteristic of exploration in organizational learning. Overall, our review indicates that agile IT teams need not just special knowledge in different topics and not only generalist understanding. Within an optimal setting, agile teams require a sufficient combination of both types of human capital. Thereby, the team should have a great diversity of skillsets to understand the customer's requirements, but also special knowledge to find the right solutions with the help of a motivated attitude to work with agile methods and react fast to moving environments.

Cooperative vs. entrepreneurial social capital: Prior research exhibits that success depends on the individual ability and willingness to work together to generate advantage for the company (Subramaniam and Youndt, 2005). Communication, shared knowledge, and trust play crucial roles for good collaboration within cross-functional teams. The challenge is to implement strong ties over all three dimensions of social capital within the team, namely structural, relational, and cognitive. Literature highlights that the individual knowledge of a particular team member should be shared among members of the agile group to learn from each other, and thus, foster group performance (Spohrer et al., 2012). Each team member should have close ties with the other team members (Pries-Heje and Pries-Heje, 2011). Agile IT teams need close cooperative ties between all members, which based on the individuals that have trust in each other to deliver exploitative learning. The entrepreneurial aspect of organizational learning is described by Sambamurthy et al. (2003) that agile IT teams have rather a strategic foresight, the ability to look ahead to changes within business environments. This describes rather weak ties to connection beyond the team. This emphasizes the entrepreneurial, exploration view of social capital. We recap that cooperative as well as entrepreneurial social capital should be implemented jointly within agile IT teams. To be ambidextrous, it is necessary to build strong networks within organizations to gain exploitation (Tushman and O'Reilly, 1996). Weak ties are important to achieve exploration in organizational learning.

Organic vs. mechanistic organizational capital: More and more teams are arranged decentralized within an organization. Consequently, new control, cultural and governance processes are required (Ågerfalk et al., 2009). Coyle et al. (2015) highlighted that a governance structure should have two-levels to be agile: a strategic level to enable communication and coordination among the company, and an individual level to foster decision making in a team base structure. Within agile projects, it is necessary to make fast decisions to react on rapid environmental changes and thus, enhance communication. Literature shows that it is important for companies to integrate cross-functional teams within their IT department to foster organizational learning and gain better performance. Prior research depicted that teams outperform when a new product is developed by a responsible team on project level within the organization (Botzenhardt et al., 2011). Lee and Xia (2010) highlighted that agile development processes should be rather organic, dynamic, evolving instead of mechanistic, predefined and static. We summarize, that organizational capital of agile IT teams is on the one hand organic within the team level where the members must work autonomously and decisions have to be made quickly. This leads to exploratory organizational learning, because the teams search for new solutions and faster time to market delivery (Coyle et al., 2015). On the other hand, agile IT teams need predefined, mechanistic reporting lines for a suitable integration within the rest of an company (Botzenhardt et al., 2011). This contributes to exploitative learning, whereas the existing knowledge is consumed. Subsequently, agile IT teams need to balance forms of organizational learning to contribute to firm success.

Static vs. dynamic technological capital: Technological capital can be either static or dynamic. A strong strand of literature studied the trade-off between agility and stability, including issues related to exploitation and exploration (March, 1991; Gibson and Birkinshaw, 2004). Exploitation is associated with alignment, the degree how far the team fulfils customers' requirements and quality expectations under given technological conditions. IT projects are often based on static and well-defined time and cost schedules (Tiwana, 2010; Cao et al., 2013). Agile IT teams have to react fast, cost efficient and flexible on customer demands and hence, exploit existing infrastructure possibilities (Fink and Neumann, 2007). Hence, the static dimension of technological capital was identified. In contrast, exploration is connected to loosely coupled systems and technologies (He and Wong, 2004; Vinekar et al., 2006). Therefore, the dynamic dimension of technological capital was found. Agile teams work widely autonomously and should be able to decide which technology they use for their daily work. The teams need full flexibility to react quickly to changing environments and this may influence the group strategy (Li and Maedche, 2012). Highly modular systems enhance the ability to respond to changing markets (Tiwana and Konsynski, 2010). This contributes to explorative learning as the team is searching for new technologies to fulfil market demands. Additionally, system development projects must be aligned and controlled with client's needs and cost schedules. To summarize, agile IT team must explore the market for new technologies and ensure that they exploit the present technological conditions so that the customers are satisfied. Hence, both forms of technological capital are necessary within agile IT teams.

5 Discussion

Our findings extend existing knowledge through a literature based overview of the settings of self-organized agile teams. Past literature primarily focused either on characteristics of agility, agile IT departments or connections of individual characteristics, for example autonomy or agile methodology use (Pries-Heje and Pries-Heje, 2011; Tripp et al., 2016). Hitherto, there is no research available that provides a broad overview regarding the characteristics of agile IT teams. Only a few researchers provide principles that go beyond single factors (Salmela et al., 2015; Tripp et al., 2016).

The results show that teams could have different settings of organizational learning. Our findings present the optimal setting of agile IT teams in case of ambidextrous organizational learning. Nevertheless, teams could also have other IC configurations in case of imbalance. Teams that have imbalance within organizational learning could lay too much weight on one type of human, social, and organizational capital. For example, if members have too much concentration on specialist knowledge (human capital) e.g. in form of adherence of deep knowledge about prior roles or job function, they will be not able to fulfil the other team roles. For agile IT teams it is a prerequisite to represent all roles of their team to achieve high performance (Tripp et al., 2016). To counteract to this phenomenon, agile IT teams could lay the focus on more generalist human capital with the aim to gain greater skill diversity (Chow and Cao, 2008). Thereby, the team might be able to reach more balance in organizational learning.

Furthermore, the various dimensions of IC could influence each other. The focus should lay on all dimensions of IC. Specialist human capital may influence cooperative social capital. For example, a barrier for agility within IT teams could be the missing attitude of individuals towards the agile approach. If individual team members are not willing to work in agile environment this might have an impact on imbalance of organizational learning. Team members with missing attitude may dislike sharing knowledge. Hence, we emphasize that teams should motivate their members to foster their attitude (organizational capital) towards agility. If team members have a high attitude towards agile approaches and methods they might be more able to build strong ties and e.g. effective knowledge sharing (social capital) (Ghobadi and Mathiassen, 2016). This might lead to a higher degree of ambidextrous organizational learning.

As we mentioned above, we identified technological capital as necessity to develop ambidextrous organizational learning within agile IT teams. We decided to determine this new dimension in order to raise awareness for this dimension because we delivered contributions in a research area of information systems. In addition, technological capital should be balanced with the dimensions of IC. For example, technological capital should be closely coordinated with organizational capital. The challenge is to exploit technological conditions to be aligned with clients cost schedules (Cao et al., 2013) as well as explore new technologies to react fast on unforeseen environmental changes (Tiwana, 2010). This might be influenced by the governance structure of IT teams and the IT department in that it is positioned. In order to achieve ambidextrous organizational capital, agile IT teams should have a great decision making autonomy but need also strong, predefined reporting lines to abide to organizations targets and orientation (Botzenhardt et al., 2011; Coyle et al., 2015). Thus, we emphasize to align the various dimensions of capital to achieve ambidextrous organizational learning.

5.1 Theoretical Implications and Further Research

Analysing prior research and synthesizing characteristics of agile teams along IC dimensions that contribute to ambidextrous organizational learning capabilities helped to investigate the described gap. As to that, the framework and assumptions provide a starting point for further investigations considering the characteristics of agile IT teams, ambidextrous organizational learning and team performance. Further research may utilize the findings to relate different IC configurations to different levels of agility. Moreover, further research utilizing the framework would enhance our understanding on how IC of IT teams relate to their outcomes (e.g., alignment with customer needs, release frequency, etc.). Last not least, further research may investigate how the dimensions of intellectual capital related to ambidextrous organizational learning of IT teams reciprocally influence each other.

The present research highlighted that nearly all identified constructs could be ordered to IC. We identified a row of constructs within human, social, and organizational capital. Furthermore, we ordered the two constructs infrastructure flexibility (Salmela et al., 2015) and modular architecture (Tiwana and Konsynski, 2010; Cao et al., 2013) to the new dimension of technological capital. These aspects were identified as crucial factors within agile team settings. Additionally, we acknowledge two forms of technological capital, it can be static or dynamic. In addition, we found out that the combination of both forms is required to enhance the agility of IT teams.

We determine technological capital as separate dimension, because we like to show the importance of technological aspects within this area of information systems research. The given set of technology should be exploited as far as possible, but to achieve best and rapid solutions new technological possibilities should be explored. Hence, we complement existing research through characteristics of technological dimensions. Therefore, the presented framework could be enriched from further research, which investigates the correlation between human, social, organizational, and technological capital to enhance the theory.

5.2 Practical Implications and Further Research

We found out that IT departments who want to be agile need to balance IC with both forms of organizational learning. Kang and Snell (2009) suggested that the several dimensions of IC could be combined to enable a coexistence of exploitative and explorative organizational learning. We identified a row of constructs of agility that we ordered to the different dimensions of IC and to the characteristic of ambidextrous organizational learning. Forms of IC can be set into relation with each other and hence, foster internal alignment. Subsequently, we exhibit that self-organized agile teams must balance exploration and exploitation organizational learning of human, social, and organizational capital as well as technological capital.

We show that efficiency within these teams could be accelerated through the implementation of a set of skills and conditions if there is a clearing of exploration and exploitation. We highlighted that IC configurations that enable IT teams to develop ambidextrous organizational learning exist as a set of screws. Furthermore, we highlighted the necessity for ambidextrous organizational learning in case of technological capital. We depicted the static and dynamic dimensions. These screws can have different settings. We explained how these settings could be adjusted to gain higher balance of organizational learning.

Further research could explore possible configuration that might be applied in practise of the presented distinctions of organizational learning. For example, different combinations of constructs have different shapes in practise. The characteristic of human, social, organizational, and technological capital might vary in various organizations. The diverse settings of constructs of agile teams might lead to different outcomes. Additionally, further research might also consider alignment of the different dimensions within IT departments and their settings (agile, hybrid, or traditional).

5.3 Limitations

Though a structured literature search with the most relevant outlets in IS was performed, research in adjunct areas such as systems engineering might be also relevant to our research question. Subsequently, further research should include other research fields, and identify and analyze papers in outlets such as the Journal of Systems and Software. Second, the proposed synthesis reflects an extension of the few available typologies to characterize agile IT teams. However, until now, the concepts are solely grounded in prior research and need further theoretical amplification. Third, the concrete manifestation of the characteristics proposed above, their interrelations as well as effects need further elaboration – theoretically and empirically.

6 Conclusion

In this paper, we present the results of a structured literature review to gain understanding on how IT teams could develop ambidextrous organizational learning capability that enables them to respond efficiently to environmental changes? As to that, we argue that ambidextrous organizational learning is contingent on the IC of agile IT teams. Hence, we build up on the multilevel model which was delivered by Kang and Snell (2009). They described that firms can enhance ambidextrous learning if the several dimensions of IC are aligned.

We found much evidence for our assumption that agile IT teams need to balance exploration and exploitation in context of IC. Hitherto, we identified IC configurations that support such ambidextrous learning. The manifestations of human, social, organizational, and technological capital should be balanced within agile IT teams to make them efficient. With the help of prior literature, we identified eleven characteristics of agile IT teams that could be ordered to the IC dimensions and enable ambidextrous organizational learning. Our findings present evidences that the characteristics of agile IT teams contribute to ambidextrous learning, which dependent on IC. The optimal setting of an agile team need to combine both types of human, social, organizational, and technological capital. Ambidextrous organizational learning can be achieved, if agile IT teams foster the simultaneous appearance of the capabilities that are necessary for exploitation and exploration.

Furthermore, we derived a new dimension namely, technological capital and determined two specificities, it could be static and dynamic. The static form of technological capital represents organizational learning in case of exploitation and the dynamic form the exploration. We contribute to prior research by synthesizing the characteristics of IC that contributes to IT teams' ability being agile. Moreover, with the help of a literature review, we developed a research model that depicts these relationships and could guide further research. This research not only offers implication for further research, but also for practice, e.g. we show how imbalance within organizational learning could be adjusted.

References

Advance IT Minnesota (2016). “Renewing the IT Curriculum: Responding to Agile, Devops, and Digital Transformation”, St. Paul, MN.

Ågerfalk, P.J., B. Fitzgerald, and S.A. Slaughter (2009). “Introduction to the Special Issue - Flexible and Distributed Information Systems Development: State of the Art and Research Challenges.” *Information Systems Research* 20 (3), pp. 317-328.

Avison, D.E. and G. Fitzgerald (2003). “Where Now for Development Methodologies?” *Communications of the ACM* 46 (1), pp. 78-82.

Balaji, S., C. Ranganathan, and T. Coleman (2011). “IT-Led Process Reengineering: How Sloan Valve Redesigned Its New Product Development Process.” *MIS Quarterly Executive* 10 (2).

Balijepally, V., R. Mahapatra, S. Nerur, and K.H. Price (2009). “Are Two Heads Better Than One for Software Development? The Productivity Paradox of Pair Programming.” *MIS Quarterly*, pp. 91-118.

Baskerville, R. and J. Pries-Heje (2004). "Short Cycle Time Systems Development." *Information Systems Journal* 14 (3), pp. 237-264.

Baumgart, R., M. Hummel, and R. Holten (2015). "Personality Traits of Scrum Roles in Agile Software Development Teams- A Qualitative Analysis", in: *European Conference on Information Systems*, Münster, Germany.

Bhatt, G., A. Emdad, N. Roberts, and V. Grover (2010). "Building and Leveraging Information in Dynamic Environments: The Role of IT Infrastructure Flexibility as Enabler of Organizational Responsiveness and Competitive Advantage." *Information & Management* 47 (7), pp. 341-349.

Botzenhardt, A., H. Meth, and A. Maedche (2011). "Cross-Functional Integration of Product Management and Product Design in Application Software Development: Exploration of Success Factors", in: *International Conference on Information Systems*, Shanghai, China.

Broadbent, M., P. Weill, and D. St. Clair (1999). "The Implications of Information Technology Infrastructure for Business Process Redesign." *MIS Quarterly*, pp. 159-182.

Bush, A.A., A. Tiwana, and A. Rai (2010). "Complementarities between Product Design Modularity and IT Infrastructure Flexibility in IT-Enabled Supply Chains." *IEEE Transactions on Engineering Management* 57 (2), pp. 240-254.

Cao, L., K. Mohan, B. Ramesh, and S. Sarkar (2013). "Adapting Funding Processes for Agile IT Projects: An Empirical Investigation." *European Journal of Information Systems* 22 (2), pp. 191-205.

Cao, L., K. Mohan, P. Xu, and B. Ramesh (2009). "A Framework for Adapting Agile Development Methodologies." *European Journal of Information Systems* 18 (4), pp. 332-343.

Choi, J., D.L. Nazareth, and H.K. Jain (2010). "Implementing Service-Oriented Architecture in Organizations." *Journal of Management Information Systems* 26 (4), pp. 253-286.

Chow, T. and D.-B. Cao (2008). "A Survey Study of Critical Success Factors in Agile Software Projects." *Journal of Systems and Software* 81 (6), pp. 961-971.

Coyle, S., K. Conboy, and T. Acton (2015). "An Exploration of the Relationship between Contribution Behaviours and the Decision Making Process in Agile Teams", in: *International Conference on Information Systems*, Fort Worth, USA.

Crossan, M.M., H.W. Lane, and R.E. White (1999). "An Organizational Learning Framework: From Intuition to Institution." *Academy of Management Review* 24 (3), pp. 522-537.

Dougherty, D. (1992). "Interpretive Barriers to Successful Product Innovation in Large Firms." *Organization Science* 3 (2), pp. 179-202.

Dybå, T. and T. Dingsøyr (2008). "Empirical Studies of Agile Software Development: A Systematic Review." *Information and Software Technology* 50 (9), pp. 833-859.

Fang, C., J. Lee, and M.A. Schilling (2010). "Balancing Exploration and Exploitation through Structural Design: The Isolation of Subgroups and Organizational Learning." *Organization Science* 21 (3), pp. 625-642.

Fink, L. and S. Neumann (2007). "Gaining Agility through IT Personnel Capabilities: The Mediating Role of IT Infrastructure Capabilities." *Journal of the Association for Information Systems* 8 (8), pp. 440-462.

Fiol, C.M. and M.A. Lyles (1985). "Organizational Learning." *Academy of Management Review* 10 (4), pp. 803-813.

Fitzgerald, B. (1998). "An Empirical Investigation into the Adoption of Systems Development Methodologies." *Information & Management* 34 (6), pp. 317-328.

Fitzgerald, B., G. Hartnett, and K. Conboy (2006). "Customising Agile Methods to Software Practices at Intel Shannon." *European Journal of Information Systems* 15 (2), pp. 200-213.

Fruhling, A. and G.-J.D. Vreede (2006). "Field Experiences with Extreme Programming: Developing an Emergency Response System." *Journal of Management Information Systems* 22 (4), pp. 39-68.

Ghobadi, S. and L. Mathiassen (2016). "Perceived Barriers to Effective Knowledge Sharing in Agile Software Teams." *Information Systems Journal* 26 (2), pp. 95-125.

Gibson, C.B. and J. Birkinshaw (2004). "The Antecedents, Consequences, and Mediating Role of Organizational Ambidexterity." *Academy of Management Journal* 47 (2), pp. 209-226.

Goh, J.C.-L., S.L. Pan, and M. Zuo (2013). "Developing the Agile IS Development Practices in Large-Scale IT Projects: The Trust-Mediated Organizational Controls and IT Project Team Capabilities Perspectives." *Journal of the Association for Information Systems* 14 (12), p. 722.

Grant, R.M. (1996). "Toward a Knowledge-Based Theory of the Firm." *Strategic Management Journal* 17 (S2), pp. 109-122.

Gupta, A.K., K.G. Smith, and C.E. Shalley (2006). "The Interplay between Exploration and Exploitation." *Academy of Management Journal* 49 (4), pp. 693-706.

Harris, M.L., R.W. Collins, and A.R. Hevner (2009). "Control of Flexible Software Development under Uncertainty." *Information Systems Research* 20 (3), pp. 400-419.

He, Z.-L. and P.-K. Wong (2004). "Exploration Vs. Exploitation: An Empirical Test of the Ambidexterity Hypothesis." *Organization Science* 15 (4), pp. 481-494.

Hovorka, D.S. and K.R. Larsen (2006). "Enabling Agile Adoption Practices through Network Organizations." *European Journal of Information Systems* 15 (2), pp. 159-168.

Hummel, M., C. Rosenkranz, and R. Holten (2013a). "Explaining the Changing Communication Paradigm of Agile Information Systems Development: A Research Model, Measurement Development and Pretest", in: *European Conference on Information Systems*, Utrecht, Netherlands.

Hummel, M., C. Rosenkranz, and R. Holten (2013b). "The Role of Communication in Agile Systems Development." *Business & Information Systems Engineering* 5 (5), pp. 343-355.

Kang, S.-C., S.S. Morris, and S.A. Snell (2007). "Relational Archetypes, Organizational Learning, and Value Creation: Extending the Human Resource Architecture." *Academy of Management Review* 32 (1), pp. 236-256.

Kang, S.C. and S.A. Snell (2009). "Intellectual Capital Architectures and Ambidextrous Learning: A Framework for Human Resource Management." *Journal of Management Studies* 46 (1), pp. 65-92.

Karahanna, E. and D.S. Preston (2013). "The Effect of Social Capital of the Relationship between the Cio and Top Management Team on Firm Performance." *Journal of Management Information Systems* 30 (1), pp. 15-56.

Krancher, O. and P. Luther (2015). "Software Development in the Cloud: Exploring the Affordances of Platform-as-a-Service", in: *International Conference on Information Systems*, Fort Worth, USA.

Kude, T., S. Bick, C. Schmidt, and A. Heinzl (2014). "Adaptation Patterns in Agile Information Systems Development Teams", in: *European Conference on Information Systems*, Tel Aviv, Israel.

Lee, G. and W. Xia (2010). "Toward Agile: An Integrated Analysis of Quantitative and Qualitative Field Data on Software Development Agility." *MIS Quarterly* 34 (1), pp. 87-114.

Lee, O.-K., V. Sambamurthy, K.H. Lim, and K.K. Wei (2015). "How Does IT Ambidexterity Impact Organizational Agility?" *Information Systems Research* 26 (2), pp. 398-417.

Li, Y. and A. Maedche (2012). "Formulating Effective Coordination Strategies in Agile Global Software Development Teams", in: *International Conference on Information Systems,*, Orlando, USA.

Lowry, P.B. and D. Wilson (2016). "Creating Agile Organizations through IT: The Influence of Internal IT Service Perceptions on IT Service Quality and IT Agility." *The Journal of Strategic Information Systems* 25 (3), pp. 211-226.

Mangalaraj, G., R. Mahapatra, and S. Nerur (2009). "Acceptance of Software Process Innovations – the Case of Extreme Programming." *European Journal of Information Systems* 18 (4), pp. 344-354.

March, J.D. (1991). "Exploration and Exploitation in Organizational Learning." *Organization Science* 2 (1), pp. 71-87.

Maruping, L.M., V. Venkatesh, and R. Agarwal (2009). "A Control Theory Perspective on Agile Methodology Use and Changing User Requirements." *Information Systems Research* 20 (3), pp. 377-399.

Matook, S. and L.M. Maruping (2014). "A Competency Model for Customer Representatives in Agile Software Development Projects." *MIS Quarterly Executive* 13 (2), pp. 77-95.

Nahapiet, J. and S. Ghoshal (1998). "Social Capital, Intellectual Capital, and the Organizational Advantage." *Academy of Management Review* 23 (2), pp. 242-266.

Nerur, S., R. Mahapatra, and G. Mangalaraj (2005). "Challenges of Migrating to Agile Methodologies." *Communications of the ACM* 48 (5), pp. 72-78.

O'Reilly, C.A. and M.L. Tushman (2008). "Ambidexterity as a Dynamic Capability: Resolving the Innovator's Dilemma." *Research in Organizational Behavior* 28, pp. 185-206.

Overhage, S. and S. Schlauderer (2012). "How Sustainable Are Agile Methodologies? Acceptance Factors and Developer Perceptions in Scrum Projects", in: *European Conference on Information Systems*, Barcelona, Spain.

Pries-Heje, L. and J. Pries-Heje (2011). "Agile & Distributed Project Management: A Case Study

Revealing Why Scrum Is Useful." In: *Proceedings of the European Conference on Information Systems*. Helsinki, Finland.

Ramesh, B., K. Mohan, and L. Cao (2012). "Ambidexterity in Agile Distributed Development: An Empirical Investigation." *Information Systems Research* 23 (2), pp. 323-339.

Rdiouat, Y., S. Bahsani, M. Lakhdissi, and A. Semma (2015). "Measuring and Improving Information Systems Agility through the Balanced Scorecard Approach." *International Journal of Computer Science Issues (IJCSI)* 12 (5), p. 58.

Salmela, H., T. Tapanainen, A. Baiyere, M. Hallanoro, and R. Galliers (2015). "IS Agility Research: An Assessment and Future Directions", in: *European Conference on Information Systems*, Münster, Germany.

Sambamurthy, V., A. Bharadwaj, and V. Grover (2003). "Shaping Agility through Digital Options: Reconceptualizing the Role of Information Technology in Contemporary Firms." *MIS Quarterly*, pp. 237-263.

Sarker, S., C.L. Munson, S. Sarker, and S. Chakraborty (2009). "Assessing the Relative Contribution of the Facets of Agility to Distributed Systems Development Success: An Analytic Hierarchy Process Approach." *European Journal of Information Systems* 18 (4), pp. 285-299.

Scheerer, A. and T. Kude (2014). "Exploring Coordination in Large-Scale Agile Software Development: A Multiteam Systems Perspective", in: *International Conference on Information Systems*, Auckland, New Zealand.

Schlauderer, S., S. Overhage, and B. Fehrenbach (2015). "Widely Used but Also Highly Valued? Acceptance Factors and Their Perceptions in Water-Scrum-Fall Projects", in: *International Conference on Information Systems*, Fort Worth, USA.

Schmidt, C., T. Kude, A. Heinzl, and S. Mithas (2014). "How Agile Practices Influence the Performance of Software Development Teams: The Role of Shared Mental Models and Backup", in: *International Conference on Information Systems* Auckland, New Zealand.

Schwartz, R.B. and M.C. Russo (2004). "How to Quickly Find Articles in the Top IS Journals." *Communications of the ACM* 47 (2), pp. 98-101.

Sommerville, I. (2009). *Software Engineering*. Boston, Massachusetts: Pearson Education, Inc.

Spohrer, K., B. Gholami, and A. Heinzl (2012). "Team Learning in Information Systems Development - A Literature Review", in: *European Conference on Information Systems*, Barcelona, Spain.

Spohrer, K., T. Kude, C.T. Schmidt, and A. Heinzl (2013). "Knowledge Creation in Information Systems Development Teams: The Role of Pair Programming and Peer Code Review", in: *European Conference on Information Systems*, Utrecht, Netherlands.

Subramaniam, M. and M.A. Youndt (2005). "The Influence of Intellectual Capital on the Types of Innovative Capabilities." *Academy of Management Journal* 48 (3), pp. 450-463.

Taylor, A. and H.R. Greve (2006). "Superman or the Fantastic Four? Knowledge Combination and Experience in Innovative Teams." *Academy of Management Journal* 49 (4), pp. 723-740.

Tiwana, A. (2010). "Systems Development Ambidexterity: Explaining the Complementary and Substitutive Roles of Formal and Informal Controls." *Journal of Management Information Systems* 27 (2), pp. 87-126.

Tiwana, A. and B. Konsynski (2010). "Complementarities between Organizational IT Architecture and Governance Structure." *Information Systems Research* 21 (2), pp. 288-304.

Torkar, R., P. Minoves, and J. Garrigós (2011). "Adopting Free/Libre/Open Source Software Practices, Techniques and Methods for Industrial Use." *Journal of the Association for Information Systems* 12 (1), p. 88.

Tripp, J.F., C. Riemenschneider, and J.B. Thatcher (2016). "Job Satisfaction in Agile Development Teams: Agile Development as Work Redesign." *Journal of the Association for Information Systems* 17 (4), p. 267.

Tushman, M.L. and C.A. O'Reilly (1996). "The Ambidextrous Organizations: Managing Evolutionary and Revolutionary Change." *California Management Review* 38 (4), pp. 8-30.

Vial, G. and S. Rivard (2015). "Understanding Agility in ISD Projects", in: *International Conference on Information Systems,* Fort Worth, USA.

Vidgen, R. and X. Wang (2009). "Coevolving Systems and the Organization of Agile Software Development." *Information Systems Research* 20 (3), pp. 355-376.

Vinekar, V., C.W. Slinkman, and S. Nerur (2006). "Can Agile and Traditional Systems Development Approaches Coexist? An Ambidextrous View." *Information Systems Management* 23 (3), pp. 31-42.

Wagner, H.-T., D. Beimborn, and T. Weitzel (2014). "How Social Capital among Information Technology and Business Units Drives Operational Alignment and IT Business Value." *Journal of Management Information Systems* 31 (1), pp. 241-272.

Wang, X., K. Conboy, and M. Pikkarainen (2012). "Assimilation of Agile Practices in Use." *Information Systems Journal* 22 (6), pp. 435-455.

Wang, Z., N. Wang, and H. Liang (2014). "Knowledge Sharing, Intellectual Capital and Firm Performance." *Management Decision* 52 (2), pp. 230-258.

Watts, S. and J.C. Henderson (2006). "Innovative IT Flimates: CIO Perspectives." *The Journal of Strategic Information Systems* 15 (2), pp. 125-151.

Webster, J. and R.T. Watson (2002). "Analyzing the Past to Prepare for the Future: Writing a Literature Review." *MIS Quarterly* 26 (2), pp. 13-23.

West, D., T. Grant, M. Gerush, and D. D'silva (2010). "Agile Development: Mainstream Adoption has Changed Agility." *Forrester Research* 2 (1), p. 41.

Wolfswinkel, J., E. Furtmueller, and C. Wilderom (2013). "Using Grounded Theory as a Method for Rigorously Reviewing Literature." *European Journal of Information Systems* 22 (1), pp. 45-55.

Yang, H., P. Antunes, and M. Tate (2016). "Towards a Unified Conceptualisation of IS Agility", in: *IEEE 20th International Conference on Computer Supported Cooperative Work in Design (CSCWD),* Nanchang, China

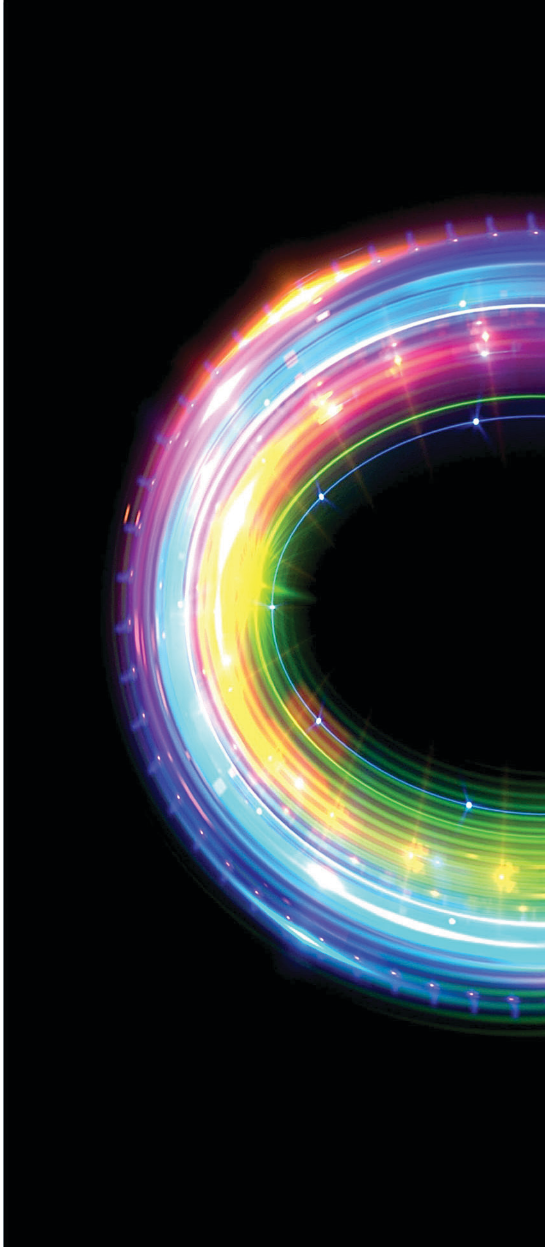
DOI:10.1145/3331138

Article development led by acmqueue
queue.acm.org

An executive crash course.

BY ANNA WIEDEMANN, NICOLE FORSGREN, MANUEL WIESCHE, HEIKO GEWALD, AND HELMUT KRCMAR

Research for Practice: The DevOps Phenomenon



A TRADITIONAL SOFTWARE company releases its flagship product maybe every few years. Each release can include hundreds of new features and improvements. Because releases are infrequent, users can grow impatient waiting for each new release and are thankful when it finally arrives.

Disappointment sets in, however, when bugs are found and features don't work as expected. Under great stress and with great turmoil, an emergency release is produced and put into production (hurried through the regular release process, often achieved by skipping tests), which has still more bugs, and the process repeats with more emergency releases, leading to more frustration, stress, and disappointment. Worse yet, new business opportunities are missed or ignored because of doubt, uncertainty, and distrust in the IT department's ability to deliver value.

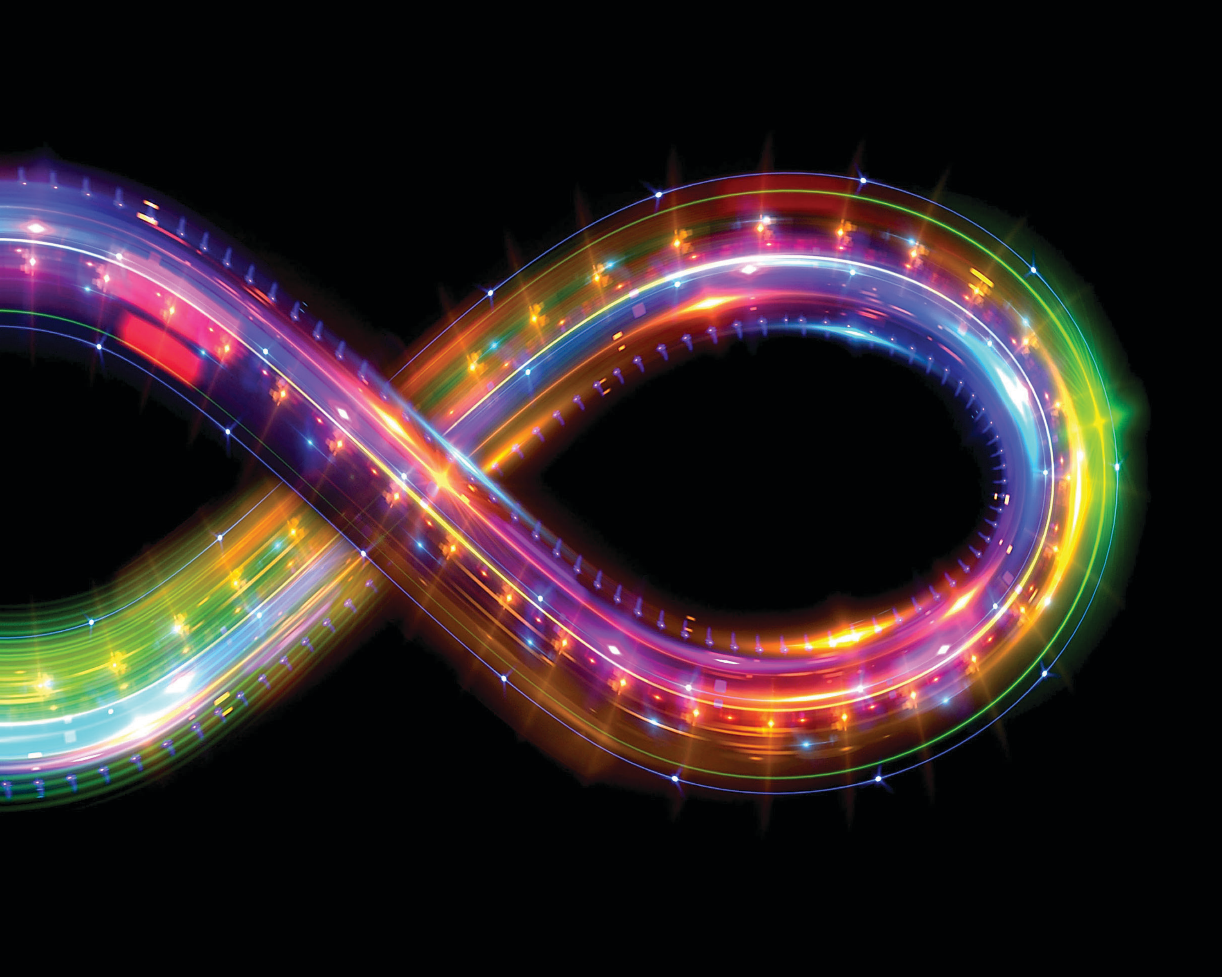
Isn't there a better way?

Such practices are a thing of the past for companies that subscribe to the DevOps method of software development and delivery. New releases are frequent: often weekly or daily. Bugs are fixed rapidly. New business opportunities are sought with gusto and confidence. New features are released, revised, and improved with rapid iterations. In one case study, a company was able to provide a new software feature every 11 seconds.[17]

Which of these software teams would you rather be? Which of these companies will win during their industry's digital transformation?

DevOps presents a strategic advantage for organizations when compared with traditional software-development methods (often called *phase-gate* or *waterfall*.[7] Leadership plays an important role during that transformation.

IMAGE BY DMITRIY RYBIN



In fact, Gartner predicts that CIOs who haven't transformed their teams' capabilities by 2020 will be displaced.[9]

Promises and Challenges for Digital Transformation

For organizations hoping to capture market share and deliver value faster (or even just deliver software more safely and securely), DevOps promises both speed and stability.[4] It has developed as a prominent phenomenon of digital transformation in modern organizations that use software to deliver value to their customers in industries including banking, retail, and even manufacturing. DevOps combines activities of software development and delivery to enhance the speed of getting new software features to customers. This leads to higher customer satisfaction and profitability, which are important outcomes at the organization level. It also leads to important team-level outcomes such as better collaboration among different departments (such as developers, testers, and IT operations) and improved work-life balance.

Executing a successful DevOps transformation isn't without its challenges. Organizations and software products vary in maturity and implementation, making transformation efforts difficult to design and deploy across teams and organizations. Most importantly, for DevOps to truly deliver value, it must include more than just tooling and automation—so simply purchasing and installing a solution is not sufficient. As outlined here, DevOps includes culture, process, and technology. Indeed, success stories abound: Companies such as Kaiser Permanente, Capital One, Target, Starbucks, and ING have adopted DevOps methods, allowing them to deliver software for key applications in just seconds.

DevOps enhances automation from applications to infrastructure provisioning. Continuous delivery supports automation and enables faster time to market and agile software development with fast feedback cycles. As the phenomenon is relatively new in practice, practitioners report on struggles with issues such as leadership and cultural transformation, implementing continuous delivery pipelines, and integrating a culture of collaboration in team settings.[8] Each of these areas would benefit from examination and guidance by formal research to test and augment the valuable experiences of those in industry.

A Move Away from Traditional Project Management

▸ DevOps methodology is marked by a change in how software delivery is treated: moving from a discrete project to an ongoing product. The traditional way of delivering software (referred to previously as *phase-gate* or *waterfall*) happens with the help of project management. In this paradigm, the project typically "ends" with the first major release of the new software or a set of new features delivered in a major incremental release. In general, the project team dissolves following a new release, and the responsibility for running the system is then "thrown over the wall" to operations. The operations team takes over responsibility for further changes and incident management. This leads to several problems:

- Developers are not responsible for running the system they built and therefore do not understand if tradeoffs appear in creating and running the system, notably in the scalability and reliability of the software. This can lead to the same problems being perpetuated in future software releases, even if they are well understood by the operations team.
- The operations team is responsible for maintaining a highly reliable and stable system. Each new line of software deployment introduces change, and therefore leads to instability. This leads to a mismatch in incentives, where accepting new software from developers introduces risk (instability) and uncertainty (because less or no visibility into the development process gives them little insight into the software they are inheriting). Even if new components are of high quality, all new code adds complexity to the system and risk that the software was not developed with scalability and reliability in mind—key factors that operations must address and support in new software, as well as existing software.
- Stakeholders upstream (that is, earlier in the development process) from the production environment, including developers and the business, are not able to receive any feedback about performance until the first complete release. This generally takes place several months after project approval. Furthermore, not all features in software deliver value, and speeding up feedback to the development team and business allows for faster iteration to refine the software. This leads to wasted time and resources on features that don't ultimately deliver value. For example, research from Microsoft shows that only one-third of well-designed features delivered value to the customer.[14]

In contrast, DevOps calls for a shift to product-based management. Practically, this means there is no more "end date" to projects, and teams instead deliver features—and therefore value—continuously. An important part of achieving this is integrating teams throughout the value stream, from development through operations; some organizations are even including business stakeholders. In this model, software is a product that is maintained as a product, with delivery and value metrics being tracked by the business continuously.

State-of-the-Art Research and Practice on DevOps

One of the biggest challenges (and complaints!) in industry is the lack of a formal definition for DevOps. Many practitioners argue that this is intentional, because it allows teams and organizations to adopt a definition that works for them. In addition, they point out that having a formal definition of agile (coded in the Agile Manifesto) hasn't solved the problem of definition sprawl, and the resulting confusion around what is truly meant when an organization says it is "going agile" still plagues the industry. This lack of understanding can be challenging, so we present some common definitions for reference here.

- "DevOps is a software development and delivery methodology that provides … increased speed and stability while delivering value to organizations."[4]
- "DevOps, whether in a situation that has operations engineers picking up development tasks or one where developers work in an operations realm, is an effort to move the two disciplines closer."[16]
- "DevOps, a compound of 'development' and 'operations,' is a software development and delivery approach designed for high velocity."[17]

And yet, these definitions focus on the outcome (value through speed and stability, via Forsgren[4]) or the foundations of the discipline (bringing together development and operations, via Roche[16] or Sebastian et al.[17] If one wants to understand the components of DevOps methods, perhaps the most common presentation of these is summarized as CALMS: culture, automation, lean, measurement, and sharing.[6,12] Here is a brief definition of each component:

- *Culture.* Integration of mutual trust, willingness to learn, continuous improvement, constant flow of information, open-mindedness to changes, and experimentation between developers and operations.
- *Automation.* Implementing deployment pipelines with a high level of automation (most notably continuous integration/continuous delivery) and comprehensive test automation.
- *Lean.* Applying lean principles such as minimization of work in progress, as well as shortening and amplification of feedback loops to identify and minimize value flow breaks to increase efficiency.
- *Measurement.* Monitoring the key system metrics such as business or transactions metrics and other key performance indicators.
- *Sharing* knowledge in the organization and across organizational boundaries. Team members should learn from each other's experiences and proactively communicate.

An Interpretation of CALMS

This article uses the five core elements of CALMS as a framework.

Culture. Working as part of a Dev Ops team requires a culture of collaboration within a cross-functional team setting. In its ideal state, DevOps uses a so-called cross-functional team, which means that groups made up of developers, testers, quality assurance professionals, and IT operations engineers all work together to develop and deliver software. In this way, they are familiar with each others' work and challenges (a common phrase to describe this is "to have empathy"), which helps them create and maintain better software. For example, because they see the challenges in maintaining scalable and reliable infrastructure encountered by operations professionals, developers write more scalable and reliable code in collaboration with operations staff.

Automation. This principle requires

a suite of DevOps tools.[2,13] The following are a few examples of available automation tooling:

- *Build.* Tools for the software development and service life cycle, including compiling code, managing dependencies, creating documentation, conducting tests, and deployments of an application in different stages.
- *Continuous integration.* Constant testing of new software components.
- *Release automation.* Packaging and deploying a software component from development across all environments to production.
- *Version control.* Managing changes to the program and collecting other information. Version control is a component of configuration management.
- *Test automation.* Using software to execute tests and repeating activities.
- *Configuration management.* Reducing production provisioning and configuration maintenance with the help of reproducing the production system on development stages.
- *Continuous delivery.* A combination of principles, practices, and patterns designed to make uninterrupted deployments.
- *Logging.* Traces are essential for application management; everything must be logged.
- *Monitoring.* Identification of infrastructure problems before the customers notice them. Monitoring storage, network traffic, memory consumption, etc.

Continuous integration and delivery reduce the cost and risk of releasing software. They do this by combining automation and good practices to consistently, reliably, and repeatably perform work (such as tests and builds) that enables fast feedback and builds quality in during the software-delivery process. This helps teams deliver features faster and more reliably and, in turn, achieve faster value delivery for the organization. The highest-performing teams are able to deploy 46 times more frequently with 2,555 times shorter lead times than low performers. Failure rates are seven times less, and they are able to recover 2,604 times faster than their lower-performing peers.[8]

Lean. "Building quality in"—referenced earlier—is a key tenet in DevOps, and a principle also found in *lean*. Applied to DevOps, this means teams look for opportunities to remove waste, leverage feedback loops, and optimize automation.

Companies such as Kaiser Permanente, Capital One, Target, Starbucks, and ING have adopted DevOps methods, allowing them to deliver software for key applications in just seconds.

Let's look at an example. DevOps teams differ in size and product responsibility. In some models, a single team conducts all software development and delivery activities, including development, testing, delivery, and maintenance. They are ultimately responsible for the complete software-delivery life cycle of the software products (and may be responsible for more than one product), delivering value to the business. Lean processes allow for quick iterations and feedback throughout the development and delivery process to improve quality and build faster and more reliable systems. Examples include working in small batches to enable fast flow through the development pipeline, limiting work in process, fixing errors as they are discovered vs. at the end, and "shifting left" on security input.

These practices may sound familiar; similar practices have driven quality and value in manufacturing. (For a great story about lean manufacturing, check out episode 561 of *This American Life*,[11] which discusses NUMMI (New United Motor Manufacturing Incorporated), the joint venture between Toyota and GM in Fremont, CA.)

Measurement is another core aspect of DevOps. The ability to monitor and observe systems is important, because software development and delivery are essentially dealing with an invisible inventory that interacts in complex ways that cannot be observed. (This is in contrast to traditional physical manufacturing systems such as an automobile assembly line, described in the NUMMI case.)

Through effective monitoring, teams are able to track, watch, measure, and debug their systems throughout the software-delivery life cycle. It should be noted that metrics are also a tool for quality assurance, and measurements from several sources should be leveraged.[9]

Sharing of knowledge and information enables successful DevOps teams and helps amplify their success. By sharing practices—both successes and failures—within teams, across the organization, and across the industry, teams benefit from the learning of others and improve faster. While others have pointed out that sharing is possible in any domain and any methodology, DevOps has adopted this as a cultural norm, and many in the industry report that the field is much more collaborative than their prior work in tech.

Internal collaboration may include work shadowing or job swapping: developers are involved in operations and maintenance activities (for example, developers may even "take the pager"), and operations engineers rotate in to development and test roles, learning essential components of design and test work. In many cases, all cross-functional team members participate in the same meetings, which gives them shared context. Cross-industry sharing often takes places at conferences, with dozens of DevOps Days and other community-organized events sprouting up around the world.

The application of these principles leads to better outcomes: for individuals (seen in reduced burnout and greater job satisfaction), for teams (seen in better software delivery outcomes and better team cultures), and for organizations (seen in improved performance in measures such as profitability, productivity, customer satisfaction, and efficiency.[7,21]

Although DevOps has been an important movement in industry for more than a decade, it has not received much attention from the academic community until recently. And while CALMS principles are not always referred to using these terms, they do appear in existing research (for example, Fitzgerald and Stol[3]).

Research Summaries

The core insights of some prior Dev Ops-related research follow:

Journal Articles

▸ Fitzgerald and Stol. *The Journal of Systems and Software*[3]

Presents a continuous software-engineering pipeline and a research agenda for different continuous processes including DevOps and BizDev (business strategy and development).

▸ Forsgren, Sabherwal, and Durcikova. *European Journal of Information Systems*[10]

Highlights the roles adopted when sharing and sourcing knowledge in a system management context, and identifies strategies for optimizing outcomes.

▸ Forsgren, Durcikova, Clay, and Wang. *Communications of the AIS*[5]

Identifies the most important system and information characteristics of tools for users who maintain systems.

▸ Dennis, Samuel, and McNamara. *Journal of Information Technology*[1]

Highlights that maintenance effort should be considered during the design phase and calls this DFM (design for maintenance). The authors present insights into how the links among documents should affect both the maintenance effort and use.

▸ Sharma and Rai. *European Journal of Information Systems*[19]

Investigates how an organization's computer-aided software engineering adoption decision is influenced by individual factors of IS leaders and technological factors.

▸ Shaft and Vessey. *Journal of Management Information Systems*[18]

Aims to examine software maintenance as interlinking comprehension and modification; the relationship between these two factors is moderated by cognitive fit.

Conferences

▸ Trigg and Bødker. *ACM Conference on Computer Supported Cooperative Work*[20]

Examines how people tailor their shared PC environment and presents an understanding of how software development tailoring can be helpful in designing systems that better fit the demands.

▸ Lwakatare et al. *Hawaii International Conference on System Sciences*[15]

Investigates key challenges of DevOps adoption in embedded system domains.

▸ Wiedemann and Wiesche. *European Conference on Information Systems*[22]

Presents an ideal skill set that DevOps team members should adopt to manage the software delivery life cycle.

Practical Challenges and Opportunities

Implementing DevOps presents several challenges. First, technology—and the resulting organizational transformation—is difficult, but strong leadership can help. As many practitioners

As many practitioners note, managing and enabling the cultural changes inside an organization can be a more difficult and important challenge than implementing technical changes.

note, managing and enabling the cultural changes inside an organization can be a more difficult and important challenge than implementing the technical changes. Good leadership is vital. One approach studied in several contexts, including DevOps, is transformational leadership; this style uses five dimensions (vision, intellectual stimulation, inspirational communication, supportive leadership, and personal recognition) to inspire and guide teams.

Second, DevOps requires a custom solution for each organization. Each context is unique, and a prescriptive approach to DevOps implementation and adoption is unlikely to be successful. Teams and organizations pose a unique set of challenges and cultural norms. Each one should adopt and adapt its own approach to achieve Dev Ops success. The adaptation of Dev Ops should include development of not only technical, but also cultural, process, and measurement practices. The work of creating a unique and seemingly ad hoc technology transformation journey is difficult and may be daunting, so many organizations look for step-by-step guides. However, these are not likely to provide solutions (beyond basic advice such as "automate your toolchain") and are usually offered by those trying to sell you something.

Third, each DevOps solution should encompass a holistic view, consisting of automation (including tools and architecture), process, and culture. In many traditional approaches, specialist knowledge in one area (for example, development) is leveraged to accomplish a task before passing it off to another group. In DevOps, a move from this high specialization to include a broad understanding of more areas is necessary. (Some call this *T-shaped knowledge*, with the top part of the "T" representing broad knowledge, while the stem of the T represents deep understanding in one area of expertise.)

This allows people to understand how their work will affect and interact with more areas of the technical stack; this often requires significant additional learning and responsibilities in the transition to DevOps. Organizations should provide training and education, and not just expect technologists to augment their learning independently. Note that while some technologists consider this expansion of responsibilities and knowledge exciting, others may push back, especially those who are just a few years from retirement and comfortable in their work roles, or who see sharing information about their roles as a risk to job security. Organizations must consider these training and cultural challenges in particular and respond accordingly. (As already noted, DevOps is about much more than just technology!)

Conclusion

DevOps is about providing guidelines for faster time to market of new software features and achieving a higher level of stability. Implementing cross-functional, product-oriented teams helps bridge the gaps between software development and operations. By ensuring their transformations include all of the principles outlined in CALMS, teams can achieve superior performance and deliver value to their organizations. DevOps is often challenging, but stories from across the industry show that many organizations have already overcome the early hurdles and plan to continue their progress, citing the value to their organizations and the benefits to their engineers.

Acknowledgments

The authors would like to thank our anonymous reviewers, whose guidance and thoughtful feedback helped us to shape and refine this article.

References

1. Dennis, A.R., Samuel, B.M. and McNamara, K. Design for maintenance: how KMS document linking decisions affect maintenance effort and use. *J. Information Technology 29*, 4 (2014), 312–326.
2. Ebert, C., Gallardo, G., Hernantes, J. and Serrano, N. DevOps. *IEEE Software 33*, 3 (2016), 94–100; https://ieeexplore.ieee.org/document/7458761.
3. Fitzgerald, B. and Stol, K.-J. Continuous software engineering: A roadmap and agenda. *J. Systems and Software 123*, (2017), 176–189.
4. Forsgren, N. DevOps delivers. *Commun. ACM 61*, 4 (Apr. 2018), 32–33; https://dl.acm.org/citation.cfm?id=3174799.
5. Forsgren, N., Durcikova, A., Clay, P.F. and Wang, X. The integrated user satisfaction model: Assessing information quality and system quality as second-order constructs in system administration. *Commun. AIS 38* (2016), 803–839.
6. Forsgren, N. and Humble, J. The role of continuous delivery in IT and organizational performance. In *Proceedings of the Western Decision Sciences Institute*, 2015
7. Forsgren, N., Humble, J. and Kim, G. *Accelerate: The Science of Lean Software and DevOps: Building and Scaling High-performing Technology Organizations.* Revolution Press, Portland, OR, 2018.
8. Forsgren, N., Humble, J. and Kim, G. Accelerate: state of DevOps report: Strategies for a new economy. DORA (DevOps Research and Assessment) and Google Cloud, 2018; https://bit.ly/2vDjPmU
9. Forsgren, N. and Kersten, M. DevOps metrics. *Commun. ACM 61*, 4 (Apr. 2018), 44-48; https://dl.acm.org/citation.cfm?id=3200906.3159169.
10. Forsgren, N., Sabherwal, R. and Durcikova, A. Knowledge exchange roles and EKR performance impact: Extending the theory of knowledge reuse. *European J. Information Systems 27*, 1 (2018), 3–21.
11. Glass, I. Episode 561, "NUMMI 2015." *This American Life*; https://www.thisamericanlife.org/561/nummi-2015.
12. Humble, J. and Molesky, J. Why enterprises must adopt DevOps to enable continuous delivery. *Cutter IT Journal 24*, 8 (2011), 6–12; https://www.cutter.com/sites/default/files/itjournal/fulltext/2011/08/itj1108.pdf.
13. Humble, J. Continuous delivery sounds great, but will It work here? *Commun. ACM 61*, 4 (Apr. 2018); 34–39; https://dl.acm.org/citation.cfm?id=3173553.
14. Kohavi, R., Crook, T., Longbotham, R., Frasca, B., Henne, R., Ferres, J.L. and Melamed, T. Online experimentation at Microsoft, Data Mining Case Studies 11, 2009.
15. Lwakatare, L.E. et al. Towards DevOps in the embedded systems domain: Why is it so hard? In *Proceedings of the Hawaii International Conference on System Sciences*, 2016.
16. Roche, J. Adopting DevOps practices in quality assurance. *Commun. ACM 56*, 11 (Nov. 2013), 38–43; https://dl.acm.org/citation.cfm?id=2524721.
17. Sebastian, I.M., Ross, J.W., Beath, C., Mocker, M., Moloney, K.G. and Fonstad, N.O. How big old companies navigate digital transformation. *MIS Q. Executive 16*, 3 (2017), 197–213.
18. Shaft, T.M. and Vessey, I. The role of cognitive fit in the relationship between software comprehension and modification. *MIS Quarterly 30*, 1 (2006), 29–55.
19. Sharma, S. and Rai, A. Adopting IS process innovations in organizations: The role of IS leaders' individual factors and technology perceptions in decision making. *European J. Information Systems 24*, 1 (2015), 23–37.
20. Trigg, R.H. and Bødker, S. From implementation to design: Tailoring and the emergence of systematization. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, 1994, 45–54; https://dl.acm.org/citation.cfm?id=192869.
21. Wiedemann, A. A new form of collaboration in IT teams—exploring the DevOps phenomenon. In *Proceedings of the Pacific Asia Conference on Information Systems*, 2017, 1–12.
22. Wiedemann, A. and Wiesche, M. Are you ready for Devops? Required skill set for Devops teams. In *Proceedings of the European Conference on Information Systems*, 2018.

Anna Wiedemann is a research assistant at Neu-Ulm University of Applied Sciences and a doctoral candidate at the Technical University of Munich (TUM), Germany.

Nicole Forsgren does research and strategy at Google Cloud and is best known as lead investigator on the largest DevOps studies to date. She has been a successful entrepreneur (with an exit to Google), professor, performance engineer, and sysadmin.

Manuel Wiesche is a postdoctoral researcher at the Chair for Information Systems, Department of Informatics at the Technical University of Munich (TUM), Germany.

Heiko Gewald is research professor of information management at Neu-Ulm University of Applied Sciences and director of the Center for Research on Service Sciences.

Helmut Krcmar is chaired professor of information systems in the Department of Informatics at the Technical University of Munich (TUM) and speaker of the directorate of fortiss GmbH, the Research Institute of the Free State of Bavaria for Software and Systems.

Integrating DevOps within IT Organizations– Key Pattern of a Case Study

Anna Wiedemann[1], Manuel Wiesche[2], Heiko Gewald[3] and Helmut Krcmar[4]

Abstract: Team-oriented, service-centric approaches for software delivery processes are becoming more and more popular. One approach that has appeared on the scene in the last decade is called DevOps. To date, little is known about how cross-functional product-oriented teams can be integrated within traditional companies. Hence, we decided to contact organizations that have already started implementing DevOps-specific processes. We talked to 34 people from 10 organizations and derived new insights into the area of DevOps. We show that there are different patterns by which DevOps can be integrated within companies—e.g. with the help of platform-oriented teams. Further, we find that teams are organized in different setups and use extended workbenches for collaboration. Our case study highlights that for successful DevOps implementation, it is important to convince every stakeholder and to work with agile coaches who can foster awareness for the necessity of DevOps.

Keywords: DevOps, Key Pattern, Characteristics, Case Study

1 Introduction

Digital transformation has led to new challenges for IT functions. A lot of companies are searching for new insights to manage their IT departments for meeting new customer requirements, because their working mode is rather reactive. Many incumbent companies are at an early stage of digital transformation and are under pressure to adapt their capabilities to develop digital strategies [PWK18; SR17].

IT departments have to work with new technologies, foster their business understanding, and focus on the delivery of new software features to customers [PWKa18]. Therefore, in recent times, service-centric IT team approaches are appearing and companies have started to implement the so-called DevOps IT teams. DevOps is a portmanteau of the words development and operations [LK16]. Literature highlights that there is no common definition of DevOps [FS17]. We define DevOps as a concept for integrating the tasks, knowledge, and skills pertaining to the planning, building, and running of activities in a single cross-functional team that is responsible for the combined development and operational tasks of one or more software service products. To quickly deliver new software features and innovations, ensure the quick handling of problems, and integrate maintenance activities, IT departments should integrate cross-functional teams rather than silo-organized IT units

[1] Neu-Ulm University of Applied Sciences, Center for Research on Service Sciences, Wileystr. 1, 89231 Neu-Ulm, anna.wiedemann@hs-neu-ulm.de

[2] Technical University of Munich, Chair for Information Systems, Boltzmannstr. 3, 85748 Garching, wiesche@in.tum.de

[3] Neu-Ulm University of Applied Sciences, Center for Research on Service Sciences, Wileystr. 1, 89231 Neu-Ulm, heiko.gewald@hs-neu-ulm.de

[4] Technical University of Munich, Chair for Information Systems, Boltzmannstr. 3, 85748 Garching, krcmar@in.tum.de

[FS17]. Aligning development and operational tasks within one team can lead to higher IT and organizational performance. High-performing DevOps IT functions are able to deploy 46 times more frequently than low-performing ones [BF16]. DevOps is a rising trend and publications predict that it will become a competitive necessity. More and more IT organizations are deciding to implement the DevOps concept [FK18]. According to a Gartner survey from 2016, about 25% of 2,000 global IT companies will work with the DevOps concept and tools in the DevOps toolchain in the future [RM15]. Hence, DevOps is being recognized as an important topic in the area of software development processes and process models. DevOps is a rising trend [WW18]. Hence, we have decided to conduct a case study with companies that are already working with DevOps principles. Taking into account the fact that the companies implement DevOps in different ways, we investigate the following research question: What are the key patterns of DevOps teams?

2 The DevOps Concept

Patrick Debois is considered as the person who branded DevOps. He held the first DevOpsDays conference in Ghent, Belgium in October 2009 [Re14]. As mentioned above, DevOps combines two words—development and operations. Key activities of DevOps are automated development, deployment, and monitoring of infrastructure within one cross-functional team. Organizations start using service-centric teams, integrating different roles and responsibilities, and breaking down historically grown silo departments with a high level of specialist knowledge. With the DevOps movement, a cultural shift toward better collaboration between developers, operations people, and quality assurance is necessary. The DevOps concept is helpful for delivering faster value to customers through timely provision of new software components, reducing problems through miscommunication, and enhancement of problem resolutions [EG16].

Prominent examples like Amazon and Google have already adopted principles of the DevOps concept and now have cycle times of new software components in seconds [SR17]. The DevOps approach enables the scaling of agility to the entire IT organization. The goal of DevOps is to enhance collaboration, automation, virtualization, and tools to bridge the activities of software development and operation [LK16]. Through DevOps, solutions are delivered to avoid interruptions between different stages of the software delivery process [FS14]. The software development lifecycle consists of planning, development, and operation tasks. DevOps helps companies to integrate the necessary speed and flexibility to deliver constant and rapid development and implementation of digital innovation. Hence, risks linked with software releases can be reduced, and feedback of new software features is received faster. In addition, agile methods can be used to manage the software development part of the team [LK16]. In agile development environments, there are different "continuous" activities, the best-known of which is continuous integration (CI). CI is defined as a process that is provoked automatically and encompasses interconnected stages like acceptance test, code validation, compliance checks, and release package development [FS14]. CI disables interruptions between the development and deployment stages of software delivery. It is very significant for the DevOps phenomenon because good collaboration between development and operation is needed [FS14]. The benefits of CI are improvements in communication, higher frequency in releases, and an enhancement in the productivity of developers [FS14; SB14]. CI can lead to other continuous activities like continuous deployment (CD) and delivery [FS17]. DevOps could help to enable these processes.

DevOps complements and extends CI and releases agile software development processes by bridging new software features quickly into production and value delivery to the customer. DevOps teams can use agile methods to manage collaboration and work within their deployment environments. Companies use DevOps for better collaboration and monitoring in order to enhance the continuous delivery of new software features and consequently foster innovation [LK16].

3 An Investigation of DevOps Teams

As mentioned above, there are some prominent examples of companies already working with DevOps approaches. However, little is known about how the DevOps concept is adopted in existing companies. Hence, we decided to contact organizations that have already adapted the DevOps approach. Therefore, we searched through the internet and social business networks (e.g. Xing), and contacted people in various companies responsible for DevOps. Thus, we were able to find 10 participants for our case study. We held two or more interviews per case. After conducting the interviews, we identified different characteristics that were adopted by each case, which we grouped to four different categories:

- **Organization of IT function:** This describes how the IT function is organized and how the DevOps approach implementation begins.
- **Implementation core and product:** This describes the DevOps team setting and service.
- **CI/CD:** This describes the degree of CI/CD pipeline implementation and the deployment rates.
- **Extended workbench:** This describes how the DevOps team is organized (i.e. with greater development or operations background) and the degree of work task organization within the DevOps team.

We present the characteristics of the DevOps setup in the similar cases in Table 1.

4 Data Collection

To answer our research question, an exploratory multiple-case-study approach is adopted for a number of reasons. The case study approach is defined as “an empirical inquiry that investigates a contemporary phenomenon in depth and within its real-life context” [Yi09, p. 18]. DevOps is quite unexplored and case studies are helpful to examine new phenomena from various perspectives [Yi09]. The advantage of case studies is that they can zoom in on real-life situations and test or develop theoretical views in relation to phenomena, as they unfold in practice [Fl06].

Case	Organization of IT function	Implementation of core and product
Case 1 (Media)	Newly founded company with an IT function with high DevOps orientation.	The team is responsible for a data service for website delivery (internal service).
Case 2 (Service)	Transformation toward DevOps for several years.	The team is responsible for a product of their website (e.g. insurance service).
Case 3 (Home)	Ad hoc decisions to reorganize the IT function with DevOps teams.	The team is responsible for an activity of the online shop (e.g. shopping basket).
Case 4 (Pet Industry)	Strategic decision to reorganize the IT functions with the help of DevOps teams.	The team is responsible for an activity of the online shop (e.g. product rating).
Case 5 (Specialized Store)	Strategic decision to integrate DevOps teams for managing their online shop.	The team is responsible for an activity of the online shop (e.g. checkout).
Case 6 (Retail 1)	New company with the idea to integrate high DevOps orientation.	The team is responsible for an app development and support (e.g. delivery service).
Case 7 (Insurance)	Integration of the DevOps approach within an existing IT function.	The team is responsible for an internal delivery platform and offers self-service to other teams (internal service).
Case 8 (Bank)	Integration of the DevOps approach within an existing IT function.	The team is responsible for running and facilitating a securities management system (internal service).
Case 9 (Retail-non-food)	New company with the idea to integrate high DevOps orientation.	The team is responsible for building and running an internal platform, which is the substructure of the online shop of the company (internal service).
Case 10 (Retail 2)	Integration of the DevOps approach within an existing IT function.	The team is responsible for building and running an internal platform, which is the substructure of the online shop of the company (internal service).

Tab. 1: Findings of the Case Study Research (1/2)

Case	CI/CD	Extended workbench
Case 1 (Media)	CD is integrated. The team is able to deploy around every four days.	The team consist of 12 software engineers located in Germany and with a nearshoring partner in East Europe and shares development and operations work.
Case 2 (Service)	Frequent releases but no optimal CI/CD.	The team consists of eight software developers and is located in Germany. The team lead is responsible for the operations parts.
Case 3 (Home)	CD is integrated. The team is able to deploy several times a day.	The team consists of five software engineers. It is located in Germany and shares development and operations work.
Case 4 (Pet Industry)	No optimal CI/CD. The team is able to deploy every two weeks.	The team consist of four developers and is supported by a quality assurance engineer. The team is located in Germany and shares development and operations work.
Case 5 (Specialized Store)	No optimal CI/CD. The team is able to deploy every two weeks.	The team consists of five software developers and operations people. It is located in Germany and shares development and operations work.
Case 6 (Retail 1)	CD is integrated. The team is able to deploy every two weeks.	The team consists of four software developers and is located in Germany. The infrastructure team is a separated unit and is responsible for operations during the day.
Case 7 (Insurance)	CD is integrated. The team is able to deploy several times a day.	The team consists of 15 software engineers allocated in Germany and Asia and shares development and operations work.
Case 8 (Bank)	No optimal CI/CD. The team is able to deploy once a week.	The team consists of 15 software operations people and is located in Germany. The developers are a separate unit.
Case 9 (Retail-non-food)	CD is integrated. The team is able to deploy several times a day.	The team consists of six software operations people and is located in Germany. The developers are in a separated unit.
Case 10 (Retail 2)	No optimal CI/CD. The team is able to deploy every two weeks.	The team consist of three software developers and is located in Germany and with a nearshoring partner in Bulgaria. The infrastructure is managed by another subsidiary company of the group.

Tab. 1: Findings of the Case Study Research (2/2)

5 Data Analysis

We applied a multiple-case-study design that enables cross-case pattern search. This method helps to examine processes and patterns over several cases to understand how similar or contrasting the results are [MH94; Yi09]. The data were analyzed with the help of “within-case analysis” as well as “cross-case-analysis”. In the within-case analysis, each team was seen as a stand-alone entity and analyzed [Yi09]. The analysis process was conducted through the lens of the key pattern. The focus was on cross-case analysis, to compare the cases and identify the patterns obtained for each case. In the present research, the coding approach emphasized by Miles and Huberman (1994) has been applied. Afterwards, in vivo coding was applied to examine each case and emergent topics or explanations. Additionally, it helps to achieve more familiarity with each case and fosters cross-case analysis [MH94; Yi09].

6 Key Findings and Discussion of the Case Study

In the following section, we present several insights achieved through our case study, as well as the key pattern identified in our research.

6.1 Insights from the DevOps Teams

To summarize, we talked with 34 persons from the aforementioned companies. All of them offer different insights into the DevOps setting. During our investigation, we recognized that there are different patterns across the several cases. Nevertheless, we found some similarities as well. Table 2 presents an illustrative example of the key findings that we identified in the different IT functions. The table presents two patterns generated with the help of our case study.

Cases 5, 6, 7, 9, and 10 have integrated a so-called platform team, which they have organized with the help of the DevOps approach. That means that an internal platform is used as the basic infrastructure—as the foundation for the online shop of the organizations. The infrastructure teams are often organized with the help of operations people who have high interest in development tasks as well. For example, they develop their own software program and write playbooks, mostly using Linux as the operating system. Hence, for them, it is a key requirement that new team members should have knowledge in Linux.

DevOps platform teams usually have operations background and undertake on-call duty outside normal business hours or developers with experience in system administration. The development teams are responsible for their services during the normal business hours. For a couple of reasons such as costs, companies may decide not to integrate on-call duty in the development teams. Very often, a lot of deployments are made during the day.

Companies start implementing some rules and regulations for deployments. For example, one case mentions that new software components are not allowed to be deployed into the production environment on Fridays, because they want that the system to be running stably during the weekend. Another case mentions that if a developer deploys new software components after the regular deployment times, the person has to be available for the DevOps platform team in the case of failure messages. The overall aim of the platform DevOps teams is to integrate self-services for the development teams. This means that developers are able to help themselves with the particular services they need (e.g. databases or firewalls). The DevOps platform team serves these services to the development teams with the help of application program interfaces (APIs).

Pattern 1	**Pattern 2**
DevOps teams have an internal platform orientation with self-service opportunity for development teams.	The IT department mainly consists of DevOps teams that are working with e.g. micro-service architecture.
Most team members have operations background and have started working with development tasks.	Most team members have software development tasks and have to learn operations tasks.
The team is mainly responsible for providing infrastructure to the development teams and supporting them.	The team is mainly responsible for service and infrastructure. If superordinate problems appear, they are supported by a support unit.

Tab. 1: Pattern of DevOps Teams

The other cases that we investigate present insights into another DevOps setup. Cases 1, 3, 4, and (partly) 2 claim that they have integrated the development and operations tasks within one team. The CTO of Case 3 mentions that it is important for the development team to be fully responsible for its service. Most of the cases are responsible for one service of an online shop (e.g. shopping basket, check-out processes). If problems appear, they have to manage it as well. The DevOps team has a high decision-making freedom regarding new technology. Furthermore, it has to guarantee that skills and knowledge necessary for managing the service are integrated within the team. For example, Case 2 mentions that it is not possible for everyone to know everything. The team should be organized so as to ensure that the service is working, even if team members are on holiday or unwell. Hence, team members should be able to stand in for each other.

IT organizations and companies organize meetings, tech-talks, and other presentations to share knowledge within and outside the company. Sharing knowledge is a key factor in the DevOps movement [BC13]. These knowledge-sharing activities are helpful because the teams learn from each other. Additionally, the teams are supported by agile coaches or disciplinarians. These people manage the teams and help them with decisions —for example, deciding whether new technologies are really necessary if the technology already exists within the company with the help of another tool. Figure 1 visualizes two examples of the different DevOps setups observed in several cases.

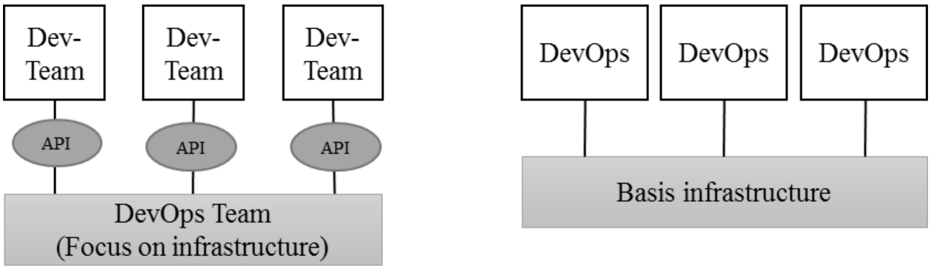


Fig. 1: Organizational Structure of DevOps

Furthermore, our findings show that the integration of DevOps presents challenges for the existing IT function and organizational structure. We recognize that the IT functions—whether completely reorganized with DevOps teams or organized as bimodal IT functions—are traditional silo units and DevOps teams coexisting.

6.2 Starting Points for DevOps

Another finding of our case study is that a lot of tradition-oriented companies are searching for ways to implement DevOps within their company. The study participants mention different starting point for the implementation of DevOps within new or existing IT functions. We summarize the following four different starting points for DevOps implementations:

- Spin-offs/subsidiary companies
- Greenfield projects
- Slow accession process
- Ad-hoc adjustment

Our findings show that different starting points for a successful implementation are possible. These opportunities should be considered by organizations trying to implemented DevOps.

6.3 Movement Towards Product Orientation

A major finding of our study regarding the organization of DevOps teams is that they do not usually work in a project setup. They are organized in the so-called product setups. IS projects usually have pre-defined project aims such as project delivery on-time and within-budget [Ki04]. Our findings suggest that there is a shift toward product-orientation—this means that there are no regular starting or ending dates for the product team. Furthermore, budget and on-time milestones were not controlled by the team in our investigation. A redesign of processes and end-to-end service responsibility is important for cross-functional DevOps teams [BRC11].

7 Conclusion for DevOps Implementation

Our findings present insights into the DevOps implementation. We depict different characteristics and types of DevOps patterns across our case-study participants. For the implementation of the DevOps concept, it is important to convince every stakeholder. We present an overview of two key patterns that we identified with the help of our case study. The first pattern consists of operations team members with a highly internal platform orientation with the aim to integrate self-services for development teams. The second pattern presents evidence that DevOps teams have high general knowledge among team members with developer backgrounds. The teams are responsible for one or more product-oriented services.

The stakeholders should be integrated into strategic decision-making processes. The integration of DevOps teams is accompanied by several challenges. The DevOps teams need more freedom for decision-making processes and higher self-responsibility for their service. Thus, traditional hierarchies should be broken down. Organizations should only give a minimum number of rules and regulation, e.g. for technology choices. Traditional silo-organized IT departments with highly specialized knowledge need to be reorganized. Cross-functional (service-centric) teams with general knowledge about the service should be integrated within the IT department.

8 Acknowledgment

This research has received funding from the German Federal Ministry of Education and Research (BMBF) under grant code 03FH005PX4. The responsibility for the content lies with the authors.

References

[BRC11] Balaji, S., Ranganathan, C., Coleman, T.: IT-Led Process Reengineering: How Sloan Value Redesigned Its New Product Development Process. MIS Quarterly Executive 10/2, S. 81–92, 2011.

[BC13] Bang, S.K., Chung, S., Choh, Y., Dupuis, M.: A Grounded Theory Analysis of Modern Web Applications: Knowledge, Skills, and Abilities for DevOps. In: Proceedings of the 2nd Annual Conference on Research in Information Technology. Orlando, S. 61–62, 2013.

[BF16] Brown, A., Forsgren, N., Humble, J., Kersten, N., Kim, G.: State of DevOps Report. Puppet+DORA, 2016.

[EG16] Ebert, C., Gallardo, G., Hernantes, J., Serrano, N.: DevOps. IEEE Software 33/3, S. 94–100, 2016.

[FS14] Fitzgerald, B., Stol, K.-J.: Continuous Software Engineering and Beyond: Trends and Challenges. In: Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering. Hyderabad, 2014.

[FS17] Fitzgerald, B., Stol, K.-J.: Continuous Software Engineering: A Roadmap and Agenda. Journal of Systems and Software 123, S. 176–189, 2017.

[Fl06] Flyvbjerg, B.: Five Misunderstandings About Case-Study Research. Qualitative Inquiry 12/2, S. 219–245, 2006.

[FK18] Forsgren, N., Kersten, M.: DevOps Metrics. Communications of the ACM 61/4, S. 44–48, 2018.

[Ki04] Kirsch, L.J. Deploying Common Systems Globally: The Dynamics of Control. Information Systems Research 15/4, S. 374–395, 2004.

[LK16] Lwakatare, L., Karvonen, T., Sauvola, T., Kuvaja, P., Holmström Olsson, H., Bosch, J., Oivo. M.: Towards DevOps in the Embedded Systems Domain: Why Is It So Hard?. In: Proceedings of the Hawaii International Conference on System Sciences, Kauai, 2016.

[MH94] Miles, M. B., Huberman, A. M.: Qualitative Data Analysis: An Expanded Sourcebook. Thousand Oaks, Sage Publications (Hrsg.), 1994.

[PWK18] Pflügler, C., Wiesche, M., Krcmar, H.: Subgroups in Agile and Traditional IT Project Teams, In: Proceedings of the Hawaii International Conference on System Sciences, Waikoloa Village, 2018.

[PWKa18] Przybilla, L., Wiesche, M., Krcmar, H.: The Influence of Agile Practices on Performance in Software Engineering Teams: A Subgroup Perspective. In: Proceedings of the ACM SIGMIS-CPR, Buffalo, 2018.

[Re14] Reed, J.P.: DevOps in Practice. Sebastopol, O'Reilly Media, Inc. (Hrsg.), 2014.

[RM15] Rivera, J., van der Meulen, R.: Gartner Says by 2016, DevOps Will Evolve from a Niche to a Mainstream Strategy Employed by 25 Percent of Global 2000 Organizations. Stamford, Gartner (Hrsg.), 2015.

[SR17] Sebastian, I.M., Ross, J.W., Beath, C., Mocker, M., Moloney, K.G., Fonstad, N.O.: How Big Old Companies Navigate Digital Transformation. MISQ Executive 16/3, S. 197–213, 2017.

[SB14] Ståhl, D., Bosch, J.: Modeling Continuous Integration Practice Differences in Industry Software Development. Journal of Systems and Software 87, S. 48–59. 2014.

[WW18] Wiedemann, A., Wiesche, M..: Are You Ready for DevOps? Required Skill Set for DevOps Teams. In: Proceedings of the European Conference on Information Systems. Portsmouth, 2018.

[Yi09] Yin, R.K.: Case Study Research: Design and Methods. Thousand Oaks, Sage Publications Inc. (Hrsg.), 2009.

ARE YOU READY FOR DEVOPS?
REQUIRED SKILL SET FOR DEVOPS TEAMS

Research paper

Wiedemann, Anna, Neu-Ulm University of Applied Sciences, Neu-Ulm, Germany, anna.wiedemann@hs-neu-ulm.de

Manuel Wiesche, Technical University of Munich, Munich, Germany, wiesche@in.tum.de

Abstract

In order to react quickly to changing environments and build a customer-centric setup, more and more organizations are deciding to work with the agile IT software development approach. But for the fast delivery of new software features in very short time, other parts of the IT department are necessary as well. Hence, the DevOps concept appears and connects development to IT operations activities within service-centric IT teams. To date, there has been very little empirical research on the skills required for the successful setup of a DevOps-oriented IT team. This study addresses this gap by conducting a multi-perspective research. We have collected data with the help of a workshop and interviews with IT experts. Seven skill categories—full-stack development, analysis, functional, decision-making, social, testing, and advisory skills—with 36 concrete skills were identified. Our study highlights that a combination of distinct development, operations, and management skills is necessary to successfully work within a DevOps team. This research explains core DevOps skill categories and provides a deeper understanding of the skill set of an ideal DevOps team setting. We describe these skills and skill categories and list their implications for research and practice.

Keywords: DevOps teams, skill set, concept of skills, agile software development, workshop, IT expert interviews

1 Introduction

Digital transformation has forced organizations to rethink their requirements for IT functions. The appearance of new digital technologies presents incumbent firms with changing opportunities on the one hand and major challenges on the other (Sebastian et al. 2017). These organizations have to think about how they can compete in the digital economy. Many old companies, which are at an early stage of digital transformation, are considering the capabilities necessary to develop themselves into digital leaders (Ross et al. 2016; Sebastian et al. 2017). IT organizations are under pressure to work with new technologies, improve their business understanding, enhance speed for delivering new products, and organize themselves to be more customer-centric (Bharadwaj et al. 2013; Jöhnk et al. 2017). Consequently, more and more organizations are deciding to implement agile IT setups to foster innovative capabilities and agility (Lee et al. 2015).

To address the current challenges and enable future-proof IT setups, it is essential for IT organizations to implement agile IT teams (Jöhnk et al. 2017). Further, digital transformation fosters the development of new capabilities within the IT function as well. Agile software development is limited to the collaboration between business and software development (Lwakatare et al. 2016; Sebastian et al. 2017; Tripp et al. 2016). For the fast delivery of new software features and to enhance business value, agility has to be scaled to other parts of the IT function that are necessary for the service delivery lifecycle as well (Fitzgerald and Stol 2017; Matook et al. 2016; Ross et al. 2016). Hence, companies are transforming their IT units and implementing service-centric IT teams—the so-called DevOps teams (Sebastian et al. 2017). DevOps is an acronym of development and operation (Lwakatare et al.

2016). DevOps combines the activities of software development and operations in one team and enables the continuous deployment of new features in short cycle times (Sebastian et al. 2017). Hence, response times for customer needs are drastically reduced through the better coordination of work (Lwakatare et al. 2016).

DevOps is a rising trend and is expected to become a competitive necessity. More and more IT organizations are deciding to implement the DevOps concept (Ross et al. 2016). According to a Gartner survey conducted in 2016, about 25% of 2,000 global IT companies will work with the DevOps concept and tools according to the DevOps toolchain (Rivera and van der Meulen 2015). Hence, DevOps is recognized as an important issue in the area of software development. Innovators in the digital area, like Amazon, have already adopted the DevOps approach and are now able to provide new codes to customers in seconds. It is clear that organizations have to develop DevOps capabilities to stay competitive through the delivery of rapid innovations (Sebastian et al. 2017). But the implementation of the DevOps concept entails extensive efforts by IT leaders to build and create new values by linking existing capabilities with novel ones (Ross et al. 2016). However, for many organizations, it is not clear which skills are needed to fulfil the different tasks that are expected from a DevOps team. Lack of skills in a project environments is one of the key factors for failure of IT projects (Keil et al. 2013). A distinct skill set is highly correlated with a successful project outcome (Napier et al. 2009); hence, it is necessary to understand the required skills of DevOps teams. Despite initial intentions to investigate forms of the DevOps concept (Lwakatare et al. 2016), there is very little empirical research on the skill set of successful DevOps teams. The goal of this study is to present an empirical skill set framework for DevOps teams. Hence, with this research, we wanted to answer the following research question: *What is the ideal skill set for successful DevOps teams?*

We conducted a case study with 10 participants as a pre-study and recognized that there is the need for a common framework about the key skill set that should be present within a DevOps team. This motivated us to set up a workshop with nine experienced IT consultants in the DevOps field, moderated by two researchers, to achieve deeper insights of the DevOps phenomenon. Subsequently, we conducted further interviews with IT consultants to verify our results. To this end, we have identified skill categories that describe the ideal DevOps team's skill set. The following sections of this paper explain the DevOps phenomenon and the concept of skills. Additionally, we focus on the presentation of the workshop findings and give a description of skill categories and their characteristics. Finally, we discuss our findings and conclude with implications for theory and practice.

2 Theoretical Background

2.1 DevOps teams

To handle the complexity of rapidly changing environments, IT organizations are deciding to implement product-oriented agile IT teams. The ability to adapt to new situations is a key requirement of high-performing software development teams (Burke et al. 2006). In changing environments, teams must be able to react fast and with precision. During the last few years, the use of agile software development methods has gained high popularity in a rising number of organizations (Tripp et al. 2016; West et al. 2010), primarily due to higher customer satisfaction ensured by the provided software as well as better software quality and higher project success (Maruping et al. 2009; Tripp et al. 2016). But research and practice highlight that a tighter collaboration between the development and the operations part of an IT function is necessary to ensure that errors are quickly fixed and the quality and resilience of the software are enhanced. DevOps can help to combine these approaches and reduce software cycles by pushing new code quickly into the production environment (Fitzgerald and Stol 2017). For the rapid delivery of new software features, innovations, quick handling of problems, and integrating maintenance activities, IT departments should use cross-functional teams rather than separated silo architectures (Fitzgerald and Stol 2014).

Prior research shows that agile software development is limited to the area of development activities. Therefore, in a lot of organizations, the agile approach does not go beyond development and customer-gain deployments of new software features except in rare cases (Lwakatare et al. 2016). One reason for this is the poor communication and collaboration between developers and operations personnel. To address this issue, the DevOps approach enables the scaling of agility to the entire company (Gotel and Leip 2007). The aim of DevOps is to foster collaboration, automation, virtualization, and tools to bridge the activities of software development and operation (Humble and Molesky 2011; Lwakatare et al. 2016). Through DevOps, solutions are delivered to avoid interruptions between different stages of the complete software delivery process (Fitzgerald and Stol 2014). The entire software development life-cycle comprises the steps of planning, development, and operation tasks. DevOps helps companies to acquire the necessary speed and flexibility to ensure the constant and rapid development and implementation of digital innovation (Ross et al. 2016). Hence, risks associated with software releases can be reduced and feedback for new software features is received faster. In addition, agile methods can be used to manage the software development part and meetings of the team (Lwakatare et al. 2016).

From a practical point of view, the state of DevOps report shows that high-performing DevOps organizations are able to deploy 200 times more frequently than low-performers. The change failure rate is three times lower than in traditional settings. Further, DevOps teams spend 22% less time on unplanned work and rework and thus are able to spend more time on new feature development (Brown et al. 2016). Software development comprises the analysis of new requirement, design and coding activities, and verification and testing, whereas software operation focuses on maintenance and software installation tasks (Fitzgerald and Stol 2014). Within a DevOps team, a very broad understanding of skills is necessary. Whereas traditional silo-oriented IT functions foster specialized knowledge in one area, e.g. departments, DevOps teams need a very broad generalist setup over all tasks of the software delivery lifecycle for which they are responsible (Rouse 2016). Hence, in cross-functional IT teams, the same activities need to be implemented for service delivery as in traditional IT functions, with separated units of specialists.

2.2 Concept of skills

Skills or competency models are used to describe the abilities, knowledge, attitudes, and traits that are essential for the effective performance of an organizational position or role (Mansfield 1996; Matook and Maruping 2014). Usually, the term competency is used synonymously to skills, but there are fine differences. Skills denote the ability to finish tasks and solve problems through the application of knowledge, whereas competencies represent the ability to use skills, knowledge, and other personal or social capabilities to assess situations or to work in professional and personal environments (Boehm et al. 2013; Peppard 2010). Skills outline the abilities and behavior needed to competently perform activities and tasks that are expected by the DevOps team (Matook and Maruping 2014). Skill models are helpful for measuring the performance of people to find out if they need personal development or need to improve effectiveness. Furthermore, they support the recruitment and staff planning process (Lucia and Lepsinger 1999; Matook and Maruping 2014). Competencies are used in different research areas (Klendauer et al. 2012). A prominent example is the PMI's Project Competence Development framework, which states that project management competencies have three components—a personal, a performance, and a knowledge component (Institute 2002; Napier et al. 2009). Furthermore, prior literature describes skill requirements for IT project managers and fosters the understanding of the several skills that lead to successful project management. Napier et al. (2009) identify nine skill categories for IT project managers, including two skill categories that have rarely been discussed before. With regard to agile software development, Matook and Maruping (2014) present a competency model for customer representatives that was derived with the help of interviews. Ten competencies are presented and grouped within three core areas. However, we found little empirical evidence on the matter of skills needed by DevOps to perform successfully.

Skill models are used for different reasons, such as for the development of a job model or adapting existing skill models for similar job profiles. When starting from the scratch, deep knowledge, e.g. from experts, is necessary for generating suitable results (Matook and Maruping 2014). In DevOps teams, little explanation of job roles exists; it is a concept for collaboration. Hence, the aim of this research is to focus on the general DevOps team's skill setup. A diverse skill set helps team members to develop new perspectives, ideas, and creative outcome (Lee and Xia 2010; Matook et al. 2016). There are two types of competencies—performance-based and attribute-based. Performance-based competencies focus on skillful practice. This type of competency depicts the job know-how of the individual (Crawford 2005; Napier et al. 2009). Competent DevOps team members can be identified with the help of effective actions that lead to successful outcome. Attribute-based competencies focus on skill-as-attributes, i.e. on the degree to which the individual has the necessary set of knowledge that a DevOps team should acquire. Attribute-based competencies are characterized by the know-what (Klendauer et al. 2012; Napier et al. 2009) of DevOps team members. Through skills-as-attributes perspectives, insights into requirements of DevOps teams for successful practice can be obtained.

The skillful-practice method addresses the behaviors, actions, and activities expected by the employers (Crawford 2005). This is helpful for the investigation of activities and interactions of effective project management. On the other hand, the skills-as-attribute method concentrates on explicit and necessary knowledge, skills, and personal characteristics (Napier et al. 2009) of DevOps team members. This method is helpful for investigating required skills based on contextual aspects, measuring the relationship between the skill set and outcome of projects. For practical purposes, it is helpful to understand which skills should be considered when a DevOps team is set up. These skills can be identified through different research methods, such as with the help of qualitative methods like expert interviews, as done in prior research (Klendauer et al. 2012; Napier et al. 2009).

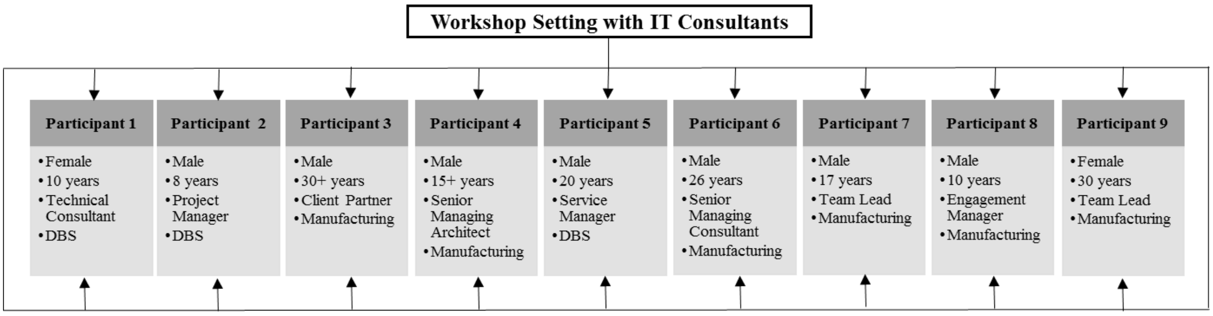
From the perspective of agile development, literature does not deliver a broader mix of skills (Matook and Maruping 2014); hence, there is a need for skill models in the area of DevOps-oriented working. For the purpose of this research, we have chosen the skill-as-attribute perspective for investigating the DevOps team members' skills. We want to provide deep insights into the skill set of DevOps teams and consider skills as mandatory for the successful performance of a DevOps team.

3 Research Method

To answer our research question, we conducted a multi-perspective research approach (Boehm et al. 2013). Therefore, we decided to undertake a comprehensive multiple-case study among companies that are already working with the DevOps approach. The qualitative instrument of the multiple-case study with 10 cases can be seen as the pre-study, which delivers the basis for a workshop with IT experts of one IT consultancy company. Workshops are a good method to study new and arising phenomena, because they allow discussion about the answers of the participants in several rounds (Boehm et al. 2013). Afterwards, three interviews with IT consultants were conducted for verification. Hence, in this paper, we use the workshop method and subsequent interviews to identify and verify the key skill set of DevOps teams.

For the case study, we contacted different firms from different industries via email and telephone. Ultimately, 10 organization agreed to participate with one DevOps team in the case study. The case study approach is defined as "an empirical inquiry that investigates a contemporary phenomenon in depth and within its real-life context" (Yin 2009, p. 18). Since the phenomenon of DevOps is quite new and unexplored (Yin 2009), the case study approach is appropriate. The advantage of case studies is that they can zoom in on real-life situations and test or develop theoretical perspectives in relation to certain phenomena as they unfold in practice (Flyvbjerg 2006). For doing so, we were able to investigate a row of skills and skill categories implemented within a DevOps team. An interview guideline not only helped to keep the interaction focused as data collection proceeded, it was also used to ensure comparability of the interview data across individuals, settings, and researchers (Maxwell 2013). Each interview lasted about 45–75 minutes and was mainly conducted through face-to-face meetings or by telephone. At least one manager (e.g. team lead) and a person concerned with daily business tasks were interviewed. Selecting both managers and team members enabled us to gain knowledge from a leadership and an operational perspective. Every interview was recorded, transcribed, and analyzed. Moreover, a comprehensive number of notes was taken during the interviews. With the help of the pre-study, we found that there is a lack of common understanding about the skills that are necessary for a DevOps team. Hence, we decided to conduct further investigation to obtain a more detailed picture about DevOps skill set for research and practice.

With the following exploratory study, we built up on the multiple-case study approach and investigated DevOps team members' construction within an ideal setting. The pre-study was the trigger for deciding to organize a workshop with IT consultant experts. We recognized that there are a lot of different skill sets in the investigated DevOps teams but a lack of a common understanding of a necessary skill set. Hence, we decided to build up on these findings and organize a workshop with IT experts. Therefore, we contacted an IT consultancy firm that is familiar with consulting projects pertaining to DevOps implementation. After a preliminary talk with two consultants, we reached an agreement about the content and identified participants for the workshop. Subsequently, we were able to conduct a personal DevOps expert workshop with nine IT consultants in October 2017. The IT consultants have different backgrounds, but are all working with DevOps topics. Our aim was to get a 360-degree view on the topic with the help of the workshop (Boehm et al. 2013). Figure 1 presents the workshop settings with the nine IT consultants. The workshop was moderated by two researchers. In sum, the figure presents the gender, working experience (years), current job role, and consultancy area of the IT consultants, namely manufacturing or digital business solutions (DBS).



Participant 1	Participant 2	Participant 3	Participant 4	Participant 5	Participant 6	Participant 7	Participant 8	Participant 9
• Female • 10 years • Technical Consultant • DBS	• Male • 8 years • Project Manager • DBS	• Male • 30+ years • Client Partner • Manufacturing	• Male • 15+ years • Senior Managing Architect • Manufacturing	• Male • 20 years • Service Manager • DBS	• Male • 26 years • Senior Managing Consultant • Manufacturing	• Male • 17 years • Team Lead • Manufacturing	• Male • 10 years • Engagement Manager • Manufacturing	• Female • 30 years • Team Lead • Manufacturing

Figure 1. Workshop participants' demographics

As far as possible, workshops should start from a blank sheet of paper; issues and ideas should emerge through collaboration between practitioners and researchers (Howard et al. 2004). Thus, the findings of the workshop can be cross-referenced against the findings of preliminary case studies and the follow-up interviews and vice versa. The workshop meeting had a duration around two and a half hours. The researchers started with a short introduction and explained the goal of the workshop. To avoid bias, the researchers focused on the process of data collection, so as to moderate the workshop without influencing the opinions of the participants (Howard et al. 2004). In the workshop, the participants were asked to elaborate the skills required by a DevOps team in an ideal setting. We asked the experts to write down on cards the key skills that they see within an ideal DevOps team. Afterwards, the experts were asked to present their findings and pinned the cards with the key skills on a wall. Every expert expanded the wall with skills that were not mentioned before and discussed the findings with the others. Skills that were mentioned by more than one participant were prioritized. The skills were discussed in the group until no new insights into the topic could be gained. Hence, we had an iterative approach where skills were discussed and the findings were extended and verified by every presentation of skills by the IT consultants. The presentation of the findings and the discussion were recorded, transcribed, and photographed. The findings of the workshop were sent to the participants and we requested them to respond with feedback and verification within one week.

After conducting the workshop, we interviewed three other IT consultants of the company (Table 1) who were not participants of the workshop and asked them for the ideal setting of skills that should be implemented within a DevOps team. The aim was to verify and extend our findings. The interview partners could not be allowed to be influenced by our prior findings. Hence, we showed them the results of the workshop in the second half of the interviews. In the first half of the interview, the participants were asked to explain the skills they see in an ideal DevOps team. Then we asked the interviewees to supplement and comment on the results of the workshop. Thereafter, we presented an overview of the achieved skill set of the workshop. These interviews provide some extensions and verifications of the workshops results. After three interviews, no new insights were given by the IT experts. All interview data and the workshop data were coded using the software NVivo 10 (Kude et al. 2014).

	Gender	Working experience	Job title	Industry
I1	Male	21 years	Managing Technical Consultant	Digital Business Solutions
I2	Male	29 years	Managing Senior Consultant	Financial Services
I3	Male	17 years	Team Lead	Manufacturing

Table 1. Interview participants' demographics

With the help of the workshop, an iterative approach for generating a skill set was used. As mentioned before, necessary skills were written down on cards and pinned to the wall. All nine participants contributed detailed information about why they felt a skill is important in a DevOps team. Furthermore, we went through transcripts of the follow-up interviews and coded them as well. We started the coding process following the guidelines given by Miles and Huberman (1994). Hence, we began with an open coding process. During the coding, the research team took notes to justify the coding section. Afterwards, the results were analyzed regarding key skills of DevOps members. Then, the research team compared its findings for each dimension to identify commonalities, relationships, and patterns. The focus was placed on the identification of the categories from the pre-study and the amplification and extension of the workshop results. Furthermore, we focused on the concrete skills that emerged during the data analysis of the expert workshop. Figure 2 depicts our research approach.

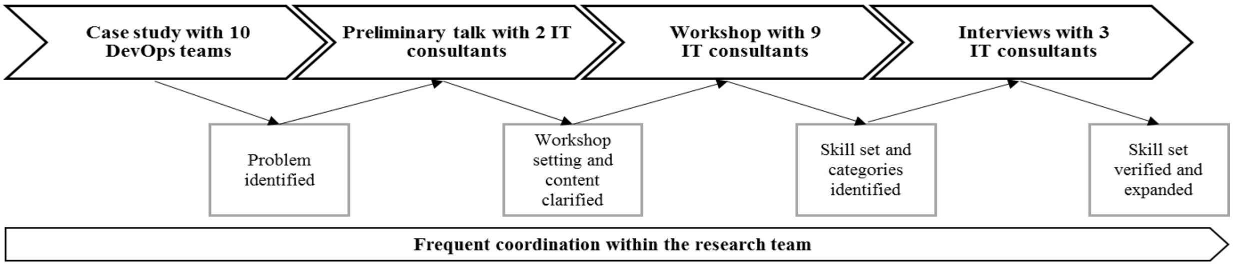

Case study with 10 DevOps teams
Preliminary talk with 2 IT consultants
Workshop with 9 IT consultants
Interviews with 3 IT consultants
Problem identified
Workshop setting and content clarified
Skill set and categories identified
Skill set verified and expanded
Frequent coordination within the research team

Figure 2. Research approach

During data analysis, the research team was able to derive key skill categories as topics for further subdivision into concrete skills to build a skill set. The concrete skills for ideal settings in DevOps teams were identified through the expert workshop and three post-processing interviews.

4 Findings

Our findings present a consolidation of the raw data with the same underlying ideas regarding skill categories, similar to the studies by Napier et al. (2009) and Klendauer et al. (2012). During the workshop, the participants mentioned some main categories or elements, and it was noted if similar detailed skills were mentioned by them, e.g. development skills. Additionally, the post-workshop interviews not only verified the skills but also automatically built core categories to sort the skills into them. After transcribing the audio data, the workshop and post-workshop interviews were coded regarding skills and core categories. Further, we extensively discussed the skills and categories in the research team, Finally, we were able to present seven core skill categories with 36 concrete skill items. The findings of our interviews for the pre-case study give insights into capabilities and skill categories of DevOps. The informants of the pre-study were unable to explain detailed skills and there was a lack of common understanding about which key skills seem to be necessary within a DevOps team. Hence, we decided to close this gap and present a common skill set for an ideal DevOps team to foster the general understanding of DevOps.

The several skill categories are now explained. The first one is **full-stack development**. We identified seven concrete skills within this category. During the pre-study, several cases highlighted that all activities of the software delivery lifecycle should be conducted by the DevOps team. The workshop participants highlighted that the comprehensive understanding of the different layers, tiers, and platforms is essential to work in a DevOps team.

Next, **analysis skills** are very important for DevOps team members. Analysis skills encompass IT operations skills (Gordon and Gordon 2002) that should now be learned by the complete cross-functional team. This is essential for monitoring, problem analysis, and management tasks. Further, all team members need skills for problem abstraction to determine where the problem could come from.

The next category we identified is **functional skills**. This category includes essential functional and consulting skills of the DevOps team. Functional skills are necessary to understand the business needs and achieve the desired benefits and goals (Liu et al. 2015). The team is now responsible for a service and thus has to manage it. This encompasses budgeting skills, work and time planning, and a deep understanding of the complete service process.

Decision-making skills are identified as another core category. DevOps members must be able to decide quickly, be self-confident, and take over responsibilities for possible failures. Decision-making is characterized by the identification of an action, the decision to take an action, determining when the action is finished, taking over the responsibility, and supporting this action (McAvoy and Butler 2009). In traditional IT organization, decision-making is a longitudinal process (Horlach et al. 2017; McAvoy et al. 2013) and now the team should be able to take over these skills for a concrete service.

The fifth category is **social skills**. These skills comprise the ability and willingness to share knowledge and communicate within the team. The IT consultants highlighted that a constructive feedback culture is essential for building a working relationship in DevOps teams (Wagner et al. 2014). The willingness of the members to actively learn from one another enhances their knowledge of technology, users, and team activities. Hence, better software quality and performance can be achieved (Spohrer et al. 2012).

Testing skills are necessary to understand the general software-testing tasks needed to provide a new software feature. For avoiding software failures, software testing is an essential activity. With the help of software testing, the correctness of the code can be proved, security issues identified, and the quality of the software guaranteed (Onita and Dhaliwal 2011). For DevOps team members, it is necessary to implement testing skills, since new software features are delivered in very short time with the help of a high degree of automation (Fitzgerald and Stol 2017).

Advisory skills were identified as the seventh core category, because the workshop participants highlighted that there must be an awareness of the commercial consequences of breaking service level agreements (SLAs) or other agreements and commitments. Thus, team members have to know these agreements. DevOps team members have different backgrounds and are not always used to working with SLAs, especially in consultant projects. If there is a failure, people must work to solve the problem as fast as possible (Trusson et al. 2014). A breach of any agreement should generally be avoided. The following table presents an overview of the identified skill categories, their description, and the identified key skills.

Skill Category	Description	Skills
Full-stack development skills	Full-stack development skills are very broad and encompasses the complete stack and interests in all software technology. This is because the DevOps team is responsible for the service delivery lifecycle. Hence, familiarity with all layers of the software and platforms is present in the DevOps team.	• Learn overarching development abilities • Know the architecture of the software system • Acquire extensive understanding of the platform • Build platforms independently • Understand non-functional requirements • Know and understand tools and supporting processes • Know and understand continuous integration/development/delivery
Analysis skills	Analysis skills mostly concern IT operation tasks like monitoring the	• Identify and resolve problems • Monitor the system operations

	system and the team's ability to comprehensively understand problem management (Gordon and Gordon 2002).	• Analyze code and network problems • Abstract problems
Functional skills	Functional skills help to understand processes from a technical as well as from a business consultant view. This encompasses, apart from the technical knowledge about the service, a structured mode of operation and consulting the customers with regard to time and budget.	• Understand the complete processes • Understand the application • Build technological knowledge • Perform tasks in a structured way • Create consulting concepts • Record and understand customer demands • Build commercial project understanding
Decision-making skills	Decision-making skills address the fast and individual decision-making process (McAvoy et al. 2013). The team has to combine them with a degree of self-organization and willingness to take responsibility.	• Take responsibility for actions and wrongdoing • Encourage appropriate decision-making • Make decisions quickly • Enable self-organization • Develop dissolving power
Social skills	Social skills pertain to the interaction within the team as well as with the customer. Thereby, the team is able to collaborate and develop working relationship for successful working (Wagner et al. 2014).	• Approve feedback ability • Communicate according to target group • Learn intercultural communication skills • Show willingness to learn new skills • Work reactively and proactively within the team • Show willingness to share knowledge • Show willingness to learn from one another
Testing skills	Testing skills ensures that the software and new features work. Testing is necessary to ensure that a software meets the specific requirements (Onita and Dhaliwal 2011).	• Understand test automation functions • Automate tests • Understand functionalities for test management
Advisory skills	Agreements within a service function help to improve collaboration. These agreements need to be understood and met (Trusson et al. 2014).	• Understand agreements • Understand the commercial impact of agreements • Comply with agreements

Table 2. Skill categories, definitions, and skills for DevOps teams

5 Discussion

This paper is one of the first to elaborate a skill model for DevOps members; hence, it contributes with insights in the area of software development and operations. In this section, we discuss the identified DevOps-related skill categories. The order of the presented skills and categories reflects their importance, as mentioned by the workshop participants and interviewees. For example, full-stack development skills were mentioned by every participant of the workshop.

Full-stack development skills: This category was mentioned as the most important by the workshop participants. These skills visualize the development perspective of a DevOps team. For the setup of a DevOps team, a comprehensive knowledge of development activity is necessary. DevOps team members need to broaden their development skills to have comprehensive knowledge about the application. Hence, skills pertaining to the different layers, tiers, and platforms are necessary. Thus, full-stack development skills are necessary *"so that it makes sense if the application at any point faces problems […] I may have some issues with the application, but it's actually because my cloud infrastructure is not working properly" (8). I*t is not sufficient for a DevOps team member to have only frontend or backend development knowledge. If a person is responsible for solving problems in a very short time, comprehensive technical knowledge is necessary. In traditional IT functions, silos for development with members who have very highly specialized knowledge in one area are used (Horlach et al. 2017).

Analysis skills: The analysis skill category encompasses IT operation activities in particular and hence, comprises the Ops of DevOps. Analysis skills with focus on problem-solving are necessary, since the DevOps team is responsible for the complete service, and if a problem appears, it has to be solved. Problem-solving skills are critical within the software delivery process and developers have to adopt this knowledge to enhance performance (Ozer and Vogel 2015). Key skills for incident and problem-solving creativity are necessary (Trusson et al. 2014). The members have to wear pagers and should be willing to do night-support activities. Within an ideal DevOps team, every member is able to contribute with analysis skills. Analysis skills to manage problems and the corresponding skills are essential for the project success. IS development projects are connected with knowledge-intensive tasks including technical and application-based domain knowledge (Lin et al. 2015). Hence, there is a need to understand the analyzing skills within the DevOps team. If team members have to do nighttime support, all members need to have the ability *“to identify from where the problem comes and analyze or solve problems so that you can at least fix them till the next day” (8),* even if it is a person with no strong IT operational background.

Functional skills: The functional skill category includes comprehensive know-how about the complete service delivery process as well as consulting skills for collaboration with the customer (Blohm et al. 2010). *“Process know-how about the development, operation, and business process is necessary” (8).* An ideal DevOps team must be clear about the functional processes. DevOps team members *“need functional knowledge about the application, or the team has to know the application from a functional perspective and should be able to assess it technically” (8).* A comprehensive knowledge of business processes and the ability to plan workloads and budget are essential in DevOps teams. Literature highlights that deeper understanding of the business functions is important for successful project collaboration. By including management and business knowledge into the team, solutions are delivered to avoid interruptions between different stages of planning, building, and running these stages (Fitzgerald and Stol 2017). Ideal DevOps teams are able to consult the customer, plan their working efforts effectively, and have a deep understanding of the development, operations, and business processes.

Decision-making skills: Decision-making skills refer to the ability of DevOps team members to make fast decisions, take responsibility if there is any failure and organize themselves within the team. The importance of decision-making skills are mentioned in existing literature with regard to agile software delivery (McAvoy and Butler 2009). Within an ideal DevOps skill setting, the team members are able to take over *“self-responsibility, enjoy decision-making, and adopt responsibility” (4).* Especially in the IT operations area, a lot of ad hoc decisions have to be made in *“stressful situations, when a major incident appears and a system is down” (5).* Usually, developers are not confronted with this fast decision-making process. Thus, organizations are under pressure to enhance the mindset and skills of developers so that they can fit in a DevOps environment. The members must *“keep calm and work with dissolving power” (I1)* when pressure from external environments appear or a major incident occurs, for example. Thus, within ideal DevOps teams, members must be able to react quickly to uncertainties and create an *“optimally organized” (I2)* team with a structured working mode.

Social skills: Social skills are necessary for interactions within the team as well as with the customer. Hence, connections between network actors, common language and code, and shared narratives, as well as the assets that create and leverage relationships should be investigated (Liu et al. 2015; Nahapiet and Ghoshal 1998). Social skills include skills like trust and knowledge-sharing, which are essential in ideal DevOps teams. The team members have to achieve a *“feedback ability, in both directions—so, to give feedback but also to accept it” (6).* This is connected with strong communications skills. Team member must be able to communicate in an audience-oriented manner. Usually, in traditional IT departments development and IT operations, people do not have a strong communication process (Horlach et al. 2017). In an ideal situation, the team members communicate, trust each other, and build a culture of communication. Furthermore, an end-to-end flow with customer demand is implemented (Fitzgerald and Stol 2017) and the DevOps team members must consider this in their communications skills as well.

Testing skills: The DevOps movement is related to a high degree of automation (Fitzgerald and Stol 2017), e.g. test automation. Continuous testing includes the automation of some test processes and defines the priority of test cases with the aim to reduce the time between implementation and detection of errors and removal of root causes (Fitzgerald and Stol 2014). Thus, ideal members of DevOps team have an understanding of software testing, which includes *"special field testing" (I3)* and know-how to automate tests and manage test automation. Software testing includes late-stage system testing on a complete and integrated system to assess compliance with the requirements (Onita and Dhaliwal 2011). Team members need testing skills because *"it is a very important topic that should be addressed early so that people can manage the tests and the automation" (5).* In summary, comprehensive testing skills are important in ideal DevOps teams.

Advisory skills: The participants of the workshop and the interviews highlight that an understanding of SLAs and their commercial impact is a key skill for DevOps teams. Not every person in the IT is familiar with SLAs or other agreements. Hence, DevOps team members have to understand the impacts agreements, because the vendor has to pay fees in case of non-compliances (Hoermann et al. 2015): *"that you understand how SLAs work. But the developers often do not have that understanding" (6).* Team members with development background are not used to working with SLAs and need an awareness for them. Ideal skilled team members of DevOps environment can effectively react to ensure that the SLA is met and no monetary damage is incurred.

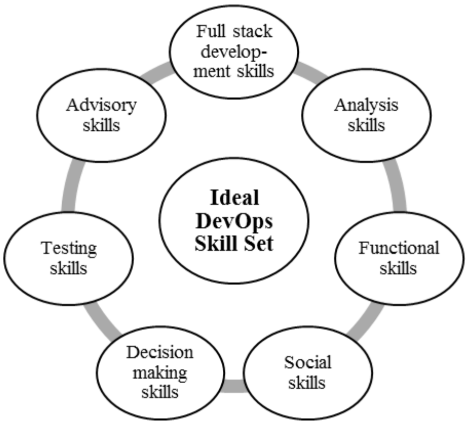


Figure 3. Skill categories of ideal DevOps teams.

In summary, we present a skill set for ideal DevOps team members with seven core categories and 36 skills. Several existing research works focus on requirements management skills (Chakraborty et al. 2010; Klendauer et al. 2012), customer representatives in agile projects (Matook and Maruping 2014), and IT project management skills (Napier et al. 2009). With the help of this paper, we contribute to existing literature by investigating the new phenomenon of DevOps. Compared to competencies in agile software development as depicted in existing research, e.g. Matook and Maruping (2014), this study shows DevOps skill sets by depicting the necessary operations skills like analysis skills and functional skills. Social-rational skills and development skills are also important in agile team settings, but the maintenance processes are usually neglected. We present a skill set for cross-functional DevOps teams that shows that skills from several areas—development, operations, and business—are important to work as a DevOps team.

IT organizations are confronted with the challenges of implementing the same skills in a very small cross-functional IT team. Therefore, a team of generalists is necessary. Team members have to leverage their knowledge in several areas in order to develop quality software (Spohrer et al. 2012). Workshop Participant 4 mentioned that *"at the current job market, I can't see where we can find such highly skilled people."* Furthermore, organizations need to build incentives that go beyond money to find well-educated people *"if you are not Facebook or Google" (1).* That means that not every company is able to pay above average salaries to recruit this highly skilled people. Hence, organizations have to rethink their development programs for employees in order to train their existing IT staff with these full-stack development skills.

During the analysis of our findings, we found that these categories can have overlaps and dependencies that should be considered. For example, decision-making skills and social skills have dependencies, because the willingness to learn new skills helps the team members to develop a strong dissolving power. Furthermore, software development and testing are major parts of the software delivery lifecycle and have to be aligned to foster business success (Onita and Dhaliwal 2011). With the help of an ideal DevOps setting, these activities are embedded in one team. Every team member should have these skills to a certain degree. Furthermore, we have defined our categories with regard to the importance of the skills, as mentioned by the workshop participants and interviewees with the help of discussions within the research team.

Furthermore, the ideal DevOps setting is dependent on several issues, one of which is governance. If IT organizations work with third-party providers or contracts, a deep understanding of SLA is necessary. If your IT functions as an internal service provider for a business section, the focus will not lie on a strong SLA process but rather on other internal agreements. Within an internal DevOps organization (without providers), *"SLAs are quite complex"* because of the high *"process character"* that should be avoided in general within an agile DevOps environment (I1). Strong process orientation is typical for traditional IT functions. Within agile-oriented IT functions, the more processes are defined, the more agility is restricted (Horlach et al. 2017). DevOps approach enables the scaling of agility to the entire company (Gotel and Leip 2007). We have included advisory skills within our research since we talked to an IT consultancy firm; SLAs and contracts were reported to be foundations of their daily work and were mentioned several times in the workshop as well as in the individual expert interviews. Hence, priorities of the skill set will be different, depending on the conditions of organization and the structure of service delivery. Furthermore, implementing concrete DevOps skills within a team is strongly dependent on the IT function and the overall organizational management.

The full-stack development, analysis, and functional skills are rarely discussed in existing literature. These skills were assessed with the highest priority. The combination of roles that are responsible for planning, building, and running within a team has been emphasized for a long time (Markus and Keil 1994). Within an ideal DevOps team, every member should be able to undertake large parts of all these activities. Since this is a new approach and a great challenge for firms, our research provides a skill set that is helpful to identify and integrate skills to start implementing the DevOps concept.

However, organizations are confronted with the challenges of digital transformation and have to adapt agile-oriented working modes like the DevOps concept. This presents a tradeoff in reality, since the necessary skill set introduces immense requirements regarding IT staff that cannot be easily found on the job market or existing silo-oriented IT function. There is only a *"short training period into a high level of complexity" (I3)* of tasks. Hence, organizations have to take great efforts to train the personnel with the presented skills to develop competitive capabilities in the area of digital transformation.

5.1 Implications for theory and practice

In summary, our research represents a major contribution to research and practice. We deliver a framework with empirically grounded insights into the skills needed by an ideal DevOps team. These identified skill categories are crucial within this ideal DevOps team setting and essential for a successful software delivery lifecycle. Hence, we deliver an initial understanding of the attribute-based competency perspectives of DevOps team members and provide a new research in this area. The present research sheds light on a DevOps team profile of an IT function that has not been systematically analyzed in depth before.

In summary, our research highlights the specific profile of DevOps teams and presents a framework that is helpful for the individual team members as well as for the organization to develop core skills. Closer collaboration is necessary between the core activities of DevOps, development, and operations. We identify DevOps-related skill categories of full-stack development, analysis, and functional skills as the most important. This shows that IT personnel with development as well as management and IT operation backgrounds have to expand their knowledge about these areas. Thus, with the help of the presented skills, we deliver concrete insights into team settings where planning, building, and running functions are integrated. We enhance the existing literature, which emphasizes the building of teams including business, development and implementations perspectives to foster performance (Markus and Keil 1994). Furthermore, the willingness to take self-responsibility and make decisions by the self-organized DevOps team and not by the management is a fundamental skill within an ideal DevOps team setting. This research focuses on the skills-as-attributes; how these skills affect the performance of DevOps teams is not in the scope of this study and should be considered by further research.

This research is the first step toward enhancing the understanding of DevOps skills, as mentioned in prior literature (Sebastian et al. 2017). We sort the identified skills into seven categories and hence provide a basis for further research. Further, we present differences and dependencies in existing research with regard to skills. Future studies can examine how these skill categories influence DevOps outcome. Moreover, interrelations between the skill categories and the several skills can be investigated. This study presents a starting point for an in-depth investigation of the field to enhance the theory.

The research presents interesting findings for practice as well. We deliver insights into how an ideal DevOps team can be set up. A set of key skills is provided by this research that can help IT managers to develop and integrate a DevOps teams. Our research provides a comprehensive overview of the skill that should be considered in DevOps team settings. We provide guidelines for the implementation of DevOps skills. Additionally, we give recommendations for designing trainings to develop highly skilled IT people. Courses for DevOps team members could be established to educate people through the organization's own IT function. This research can also be used to supervise or control project in with the DevOps approach is already used. This paper shows that practitioners increasingly need to be made aware of the DevOps concept and implement trainings and efforts to stay up-to-date. This research presents a comprehensive skill set in the case of ideal DevOps teams setup. Since these high-skilled IT people are not easy to find or train, further research should investigate how far every team member should be able to assist in every task. Furthermore, studies should investigate how DevOps teams can be supported through the IT function. Even if the team should work autonomously, there are interdependencies with other teams or the IT function. The identification of cross-sectional collaborations with other DevOps teams or with a traditional IT department is helpful for implementing DevOps teams.

5.2 Limitations

While interpreting the results, some limitations should be considered. This research focuses on the perception of IT consultants of one company. These consultants have worked in different projects with various levels of work experience. A large part of this research is based on the results of this workshop, which reduces the generalizability of our research. Further research should broaden this perspective by including a complementary study with more participants to substantiate the findings or find an additional company with a similar setting. Moreover, views of other stakeholders and internal DevOps teams might be interesting and uncover other characteristics of the skill set as well and should be investigated in further research. Furthermore, we present a skill set of an ideal DevOps setting, independent of the structure, industry, management, etc. In real-life context, there might appear some differences in the priority and characteristics of the several skills. Further, this research focuses on the IT perspective because we only talked to IT people. Agile development and DevOps team settings have a high impact on customers and business; hence, the research should be expanded through a business view as well.

6 Conclusion

This paper contributes to the understanding of skills of an ideal DevOps team. We used the concept of competencies and skills and identify attribute-based competencies that are characterized by the know-what (Klendauer et al. 2012; Napier et al. 2009) of DevOps team members. With the help of a multi-perspective research approach that comprises a case study research, workshops, and interviews for verifications, we were able to derive an ideal skill set for DevOps teams. Our findings show the importance of skills and skill categories to build effective and successful DevOps team. Since there exists less common view about DevOps key skills, we follow the path of prior literature (Fitzgerald and Stol 2017; Sebastian et al. 2017) to further investigate this phenomenon. This research enriches the understanding of several skills that were less defined in prior research. We identify seven core categories and report that three of them in particular—full-stack development, analysis, and functional skills—and above all the interaction between them, have not been widely discussed in prior IS literature.

Moreover, by analyzing and prioritizing our findings, this study highlights three major skill topics. This shows that a combination of strong development, operations, and management skills is necessary to successfully work within a DevOps team. Our findings expand existing literature in the field of skill as well as in the area of digital transformation—to be more concrete, DevOps and agile IT teams—and give practical implications as well. Existing research mainly focuses on agile and project management skills. The findings deliver a strong foundation for further research opportunities about the DevOps concept, possible relationships among the skill categories, several skills, and the effects of performance and outcomes of DevOps teams.

7 Acknowledgment

This research has received funding from the German Federal Ministry of Education and Research (BMBF) under grant code 03FH005PX4. The responsibility for the content lies with the authors.

References

Bharadwaj, A., El Sawy, O. A., Pavlou, P. A., and Venkatraman, N. 2013. "Digital Business Strategy: Toward a Next Generation of Insights," MIS Quarterly (37:2), pp. 471–482.

Blohm, I., Bretschneider, U., Leimeister, J. M., and Krcmar, H. 2010. "Does Collaboration among Participants Lead to Better Ideas in IT-Based Idea Competitions? An Empirical Investigation," in: Hawaii International Conference on System Sciences. Kauai, USA.

Boehm, M., Stolze, C., and Thomas, O. 2013. "Teaching the Chief Information Officers: An Assessment of the Interrelations within Their Skill Set," in: Wirtschaftsinformatik. Leipzig, Germany.

Brown, A., Forsgren, N., Humble, J., Kersten, N., and Kim, G. 2016. "State of DevOps Report," Puppet + DORA.

Burke, C. S., Stagl, K. C., Salas, E., Pierce, L., and Kendall, D. 2006. "Understanding Team Adaptation: A Conceptual Analysis and Model," Journal of Applied Psychology (91:6), p. 1189.

Chakraborty, S., Sarker, S., and Sarker, S. 2010. "An Exploration into the Process of Requirements Elicitation: A Grounded Approach," Journal of the Association for Information Systems (11:4), p. 212.

Crawford, L. 2005. "Senior Management Perceptions of Project Management Competence," International Journal of Project Management (23:1), pp. 7-16.

Fitzgerald, B., and Stol, K.-J. 2014. "Continuous Software Engineering and Beyond: Trends and Challenges," in: International Workshop on Rapid Continuous Software Engineering. ACM, pp. 1–9

Fitzgerald, B., and Stol, K.-J. 2017. "Continuous Software Engineering: A Roadmap and Agenda," Journal of Systems and Software (123), pp. 176–189.

Flyvbjerg, B. 2006. "Five Misunderstandings About Case-Study Research," Qualitative Inquiry (12:2), pp. 219–245.

Gordon, J. R., and Gordon, S. R. 2002. "Information Technology Service Delivery: An International Comparison," Information Systems Management (19:1), pp. 62–70.

Gotel, O., and Leip, D. 2007. "Agile Software Development Meets Corporate Deployment Procedures: Stretching the Agile Envelope," Agile Processes in Software Engineering and Extreme Programming), pp. 24–27.

Hoermann, S., Hlavka, T., Schermann, M., and Krcmar, H. 2015. "Determinants of Vendor Profitability in Two Contractual Regimes: An Empirical Analysis of Enterprise Resource Planning Projects," Journal of Information Technology (30:4), pp. 325-336.

Horlach, B., Drews, P., Schirmer, I., and Boehmann, T. 2017. "Increasing the Agility of IT Delivery: Five Types of Bimodal IT Organization," in: Hawaii International Conference on System Sciences. Waikaola Village, USA:

Howard, M., Vidgen, R., and Powell, P. 2004. "Exploring Industry Dynamics in E-Procurement: Sense Making by Collaborative Investigation," in: European Conference on Information Systems. Turku, Finland.

Humble, J., and Molesky, J. 2011. "Why Enterprises Must Adopt DevOps to Enable Continuous Delivery," Cutter IT Journal (24:8).

Institute, P. M. 2002. Project Manager Competency Development Framework. Newtown Square, USA: Project Management Institute.

Jöhnk, J., Röglinger, M., Thimmel, M., and Urbach, N. 2017. "How to Implement Agile IT Setups: A Taxonomy of Design Options," in: European Conference on Information Systems. Guimarães, Portugal:

Keil, M., Lee, H. K., and Deng, T. 2013. "Understanding the Most Critical Skills for Managing IT Projects: A Delphi Study of IT Project Managers," Information & Management (50:7), pp. 398–414.

Klendauer, R., Berkovich, M., Gelvin, R., Leimeister, J. M., and Krcmar, H. 2012. "Towards a Competency Model for Requirements Analysts," Information Systems Journal (22:6), pp. 475–503.

Kude, T., Bick, S., Schmidt, C., and Heinzl, A. 2014. "Adaptation Patterns in Agile Information Systems Development Teams," in: European Conference on Information Systems. Tel Aviv, Israel:

Lee, G., and Xia, W. 2010. "Toward Agile: An Integrated Analysis of Quantitative and Qualitative Field Data on Software Development Agility," MIS Quarterly (34:1), pp. 87–114.

Lee, O.-K., Sambamurthy, V., Lim, K. H., and Wei, K. K. 2015. "How Does IT Ambidexterity Impact Organizational Agility?," Information Systems Research (26:2), pp. 398–417.

Lin, T.-C., Chen, C.-M., Hsu, J. S.-C., and Fu, T.-W. 2015. "The Impact of Team Knowledge on Problem Solving Competence in Information Systems Development Team," International Journal of Project Management (33:8), pp. 1692–1703.

Liu, G. H., Wang, E., and Chua, C. E. H. 2015. "Leveraging Social Capital to Obtain Top Management Support in Complex, Cross-Functional IT Projects," Journal of the Association for Information Systems (16:8), pp. 707–737.

Lucia, A. D., and Lepsinger, R. 1999. Art and Science of Competency Models. San Francisco, USA. Jossey-Bass.

Lwakatare, L. E., Karvonen, T., Sauvola, T., Kuvaja, P., Olsson, H. H., Bosch, J., and Oivo, M. 2016. "Towards DevOps in the Embedded Systems Domain: Why Is It So Hard?," in: Hawaii International Conference on System Sciences Kauai, USA.

Mansfield, R. S. 1996. "Building Competency Models: Approaches for HR Professionals," Human Resource Management (35:1), pp. 7–18.

Markus, M. L., and Keil, M. 1994. "If We Build It, They Will Come: Designing Information Systems That People Want to Use," Sloan Management Review (35:4), pp. 11–35.

Maruping, L. M., Venkatesh, V., and Agarwal, R. 2009. "A Control Theory Perspective on Agile Methodology Use and Changing User Requirements," Information Systems Research (20:3), pp. 377–399.

Matook, S., and Maruping, L. M. 2014. "A Competency Model for Customer Representatives in Agile Software Development Projects," MIS Quarterly Executive (13:2), pp. 77–95.

Matook, S., Soltani, S., and Maruping, L. 2016. "Self-Organization in Agile ISD Teams and the Influence on Exploration and Exploitation," in: International Conference on Information Systems. Dublin, Ireland.

McAvoy, J., and Butler, T. 2009. "The Role of Project Management in Ineffective Decision Making within Agile Software Development Projects," European Journal of Information Systems (18:4), pp. 372–383.

McAvoy, J., Nagle, T., and Sammon, D. 2013. "Using Mindfulness to Examine Isd Agility," Information Systems Journal (23:2), pp. 155–172.

Miles, M. B., and Huberman, A. M. 1994. Qualitative Data Analysis: An Expanded Sourcebook. Thousands Oaks: Sage Publications.

Nahapiet, J., and Ghoshal, S. 1998. "Social Capital, Intellectual Capital, and the Organizational Advantage," Academy of Management Review (23:2), pp. 242–266.

Napier, N. P., Keil, M., and Tan, F. B. 2009. "IT Project Managers' Construction of Successful Project Management Practice: A Repertory Grid Investigation," Information Systems Journal (19:3), pp. 255–282.

Onita, C., and Dhaliwal, J. 2011. "Alignment within the Corporate IT Unit: An Analysis of Software Testing and Development," European Journal of Information Systems (20:1), pp. 48–68.

Ozer, M., and Vogel, D. 2015. "Contextualized Relationship between Knowledge Sharing and Performance in Software Development," Journal of Management Information Systems (32:2), pp. 134–161.

Peppard, J. 2010. "Unlocking the Performance of the Chief Information Officer (CIO)," California Management Review (52:4), pp. 73–99.

Rivera, J., and van der Meulen, R. 2015. "Gartner Says by 2016, DevOps Will Evolve from a Niche to a Mainstream Strategy Employed by 25 Percent of Global 2000 Organizations," Gartner. (ed.). Stamford, USA, from https://www.gartner.com/newsroom/id/2999017.

Ross, J. W., Sebastian, I., Beath, C., Mocker, M., Moloney, K., and Fonstad, N. 2016. "Designing and Executing Digital Strategies," in: International Conference on Information Systems. Dublin, Ireland.

Rouse, M. 2016. "DevOps," in: TechTarget. from http://searchitoperations.techtarget.com/definition/DevOps.

Sebastian, I. M., Ross, J. W., Beath, C., Mocker, M., Moloney, K. G., and Fonstad, N. O. 2017. "How Big Old Companies Navigate Digital Transformation," MISQ Executive (16:3), pp. 197-213.

Spohrer, K., Gholami, B., and Heinzl, A. 2012. "Team Learning in Information Systems Development-a Literature Review," in: European Conference on Information Systems. Barcelona, Spain.

Tripp, J. F., Riemenschneider, C., and Thatcher, J. B. 2016. "Job Satisfaction in Agile Development Teams: Agile Development as Work Redesign," Journal of the Association for Information Systems (17:4), pp. 267–307.

Trusson, C. R., Doherty, N. F., and Hislop, D. 2014. "Knowledge Sharing Using IT Service Management Tools: Conflicting Discourses and Incompatible Practices," Information Systems Journal (24:4), pp. 347–371.

Wagner, H.-T., Beimborn, D., and 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), pp. 241–272.

West, D., Grant, T., Gerush, M., and D'silva, D. 2010. "Agile Development: Mainstream Adoption Has Changed Agility," Forrester Research (2:1).

Yin, R. K. 2009. Case Study Research: Design and Methods. Thousand Oaks, CA: SAGE.

Implementing the Planning Process within DevOps Teams to Achieve Continuous Innovation

Anna Wiedemann
Neu-Ulm University of Applied Sciences
Center for Research on Service Sciences (CROSS)
anna.wiedemann@hs-neu-ulm.de

Manuel Wiesche
Technical University of Munich
Chair for Information Systems
wiesche@in.tum.de

Heiko Gewald
Neu-Ulm University of Applied Sciences
Center for Research on Service Sciences (CROSS)
Heiko.gewald@hs-neu-ulm.de

Helmut Krcmar
Technical University of Munich
Chair for Information Systems
krcmar@in.tum.de

Abstract

Integrating business capabilities into software development projects is still a major challenge for organizations. New ways of working are appearing in response to react to novel market places. Hence, there are more and more business managers with good IT knowledge; thus, software developers need to understand business processes. Hence, the relationship between software development, operations, and business strategy needs to be enhanced. For collecting customer perspectives in IT projects, new approaches like DevOps and BizDevOps are being used. The customer view can be integrated within software development teams through the planning processes. Our findings show that continuous innovation mechanisms are connected with the planning of customer requirements. We present planning scalability, security, and quality as rich descriptions of continuous innovation. Furthermore, we present core categories of how the customer perspectives can be integrated within a DevOps team and insights on how planning areas influence the continuous innovation mechanisms.

1. Introduction

Business departments are working in new ways develop and explore new markets around the world. Organizations are under pressure to integrate organizational agility and respond quickly to changing customer demands. Hence, agility is a major concern in the current business world [1]. It can be supported through Information Technology (IT) [2]. A number of trends have appeared in research and practices. For example, the concept of DevOps (Development and Operations) describes the continuous collaboration of software development and operational activities to quickly provide new software components to the customer [3, 4].

The gap between business managers and software developers is a well-known problem [5] because building business capabilities is still a great challenge for IT managers [6]. IT employees like developers are often very technical and tool-oriented; they search for the easiest technical solution to problems [7]. In the past, usually IT employees had the power to make decisions regarding the management of IT projects. But this attitude has changed because more and more business people have very good knowledge about technology. For closer cooperation, technical employees should work on their business knowledge and vice versa [8].

Existing research identifies the need for a closer connection between business managers and IT employees [9]. In practice, many software developers understand the need and are willing to collaborate with the business to make strategic decisions. It is suggested that stakeholders should be integrated into the software development process at a very early stage of the software delivery lifecycle (SDLC) [3, 8]. This integration could be achieved through planning processes. Continuous planning is an important topic in recent publications [9]. In the past, planning was often combined in annual financial cycles in traditional software development projects

URI: https://hdl.handle.net/10125/60138
ISBN: 978-0-9981331-2-6

with very few software releases [10]. A failure in the traditional planning cycle might have necessitated another planning cycle to resolve the problem. While annual planning cycles are not unusual in traditional project settings, continuous planning is considered a key prerequisite in the context of agile development and for delivering fast and new software [8, 11, 12].

Existing research highlights that a lack of an efficient IT architecture may hinder enterprise agility. Monolithic IT architectures are critical for firms when adjustments to processes are necessary in response to changing demands. However, high costs may be incurred when the organization wants to integrate a new strategy, for example [13]. Hence, to achieve enterprise agility, DevOps could be a suitable way of breaking down software monoliths into smaller services, where the responsibilities lie with one cross-functional team [18].

The relationship between software development and business strategy needs to be continuous. Literature labels this relationship as “BizDev” [8]. Existing research on information systems (IS), management, and software engineering calls for further investigation of this phenomenon [3, 8]. Continuous planning enables business and IT to work closely together in a “BizDevOps” environment [3, 14]. For enhancing the relations of business managers and IT employees, further research is necessary. The aim of this research is to determine *how an understanding of the relationship between customer demands and the DevOps approach can be achieved to enhance continuous innovations*.

We begin with a short introduction of the related literature and present the concepts of BizDevOps, enterprise agility, and continuous innovation. Then, we present our research method and describe our research approach. With the help of the case analysis, we present rich descriptions [15] of the results. Finally, we discuss our findings and conclude the paper.

2. Related Literature

2.1 The DevOps and BizDevOps Concept

To achieve a higher success rate of software development projects, a team should integrate skills and broad knowledge about the complete SDLC [4, 16]. For this, the DevOps approach could be a suitable solution, because project team members are responsible for the complete SDLC from planning to operations [17, 18]. DevOps is a new technological trend that presents new challenges for organizations [19].

Business strategy and planning tasks provide challenges for the collaboration of business and IT. According to existing literature, the term BizDev implies the necessity for continuous integration and improvement between business strategy and software development. Hence, BizDev complements the DevOps concept [8, 20]. The importance of closer collaboration between business and IT arises from the short cycles of feedback from customers, which are implemented with the help of agile project management methods [21, 22]. Furthermore, more and more business employees act as proxies in the role of agile coaches or product owners (PO) in IT projects. To meet and satisfy customers demands, it is essential for the software engineering flow to have a tight connection between business, development and operations [8]. Continuous planning is a major capability for managing systems [3], as done by DevOps teams. Hence, combining BizDev and DevOps to form BizDevOps will foster the collaboration between business and IT. Integrating professional experts into DevOps teams is a key to achieving BizDevOps.

BizDevOps is defined as the integration of domain experts within DevOps teams. A major advantage is that the tighter connection between planning and execution leads to continuous planning [8]. Hence, customer demands can be satisfied faster and the team can react quickly to changing environments.

Organizations have to rapidly adapt to react to new customer demands [23] and build DevOps and BizDevOps capabilities in order to stay competitive [19]. The reasons are higher customer satisfaction with the provided software, as well as better software quality and higher project success [24]. A tighter collaboration between development and the operations part of an IT function is necessary to ensure that errors are quickly fixed and the quality and resilience of the software are enhanced. Nowadays, it is essential to develop innovative capabilities to react to digital disruptions [25].

In traditional IT functions, business managers are responsible for planning and prioritizing the processes. Furthermore, organizations centralize highly specialized IT staff in so-called silo units in order to build new software features using sequential development methods like “waterfall development.” Afterwards, there is a long time before the software features are implemented and run by the operations IT unit. The complete process has strong dependencies on the business manager [5, 26]. Through the DevOps concept, solutions are delivered to avoid interruptions at different stages of planning, building, and running. Since the SDLC includes the steps these tasks, a tighter collaboration between planning, executing, and operating is enabled [3].

Using the DevOps concept, organizations are able to release new software features frequently and automatically [27]. Hence, risks linked to software releases can be reduced and feedback for new software features is received faster [28].

2.2 Enterprise Agility and Continuous Innovation

IS literature provides broad knowledge about enterprise and IT agility but lacks understanding of how a closer connection and flow between business and development and between business and operations can be achieved [13]. In times of uncertainty regarding planning processes in short cycle developments, agility concepts are necessary. A suitable use of IT is a key leverage factor for organizational agility [29]. Enterprise agility is defined as *"the ability of firms to sense environmental change and respond readily"* (Overby, Bharadwaj and Sambamurthy [13] p. 121). Literature highlights that a network based on trust and commitment with blurred boundaries is essential for a relationship between business and IT. A competitive advantage can be gained through better coordination, management, and structuring of relationships with stakeholders and a more agile-oriented collaboration with customers [30]. Agile software development methods could help to enhance this relationship. IT projects with short time system development enable faster delivery of innovations to the customers [29]. Existing literature states that the combining of business and technology alignment can be achieved and can supports the business cycle, deliver major benefits, and provide innovations [1].

Continuous innovation is defined as a sustainable process that supports responsiveness to new requirements and changing market demands throughout the SDLC [8, 31]. In the business context, innovations are combined with new ideas, which are transformed to achieve value for business. Continuous innovation is most widely used in the area of software development through concepts like DevOps. Thereby, early customer feedback to new software deployments can be obtained [8]. Furthermore, planning is a key prerequisite for continuous innovation. Adequate planning processes are very important for avoiding failures in the development processes. Continuous innovation helps processes to react to new market demands across the entire SDLC of planning, building, and running software [3, 8]. The BizDev approach recognizes this issue and tries to tighten the relationships between business strategies and software development. The PO is responsible for the business contact. This is emphasized by agile software development methods like scrum as the first step toward the BizDev direction.

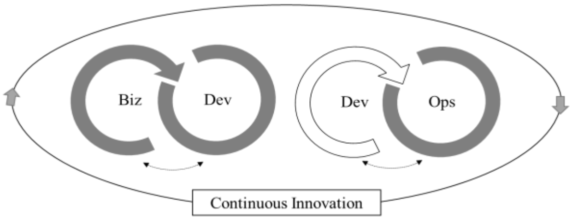


Figure 1. BizDevOps and continuous innovation

3. Research Design

We conducted a multiple-case study to analyze the flow between business stakeholder demands, software developers, and operations. Since BizDev and DevOps studies are neglected in existing literature, our aim is to provide rich descriptions with the help of grounded theory through a multiple-case study research [15, 32]. In this section, we describe our exploratory research design and approach.

The cases considered in our study have their headquarters in Germany. A case study approach is defined as *"an empirical inquiry that investigates a contemporary phenomenon in depth and within its real-life context*" (Yin [33] p. 18). The present paper is among the first studies to investigate the BizDevOps phenomenon [33]. The advantage of case study research is that it can examine real-life situations and test or develop theoretical perspectives in relation to the considered phenomena as they unfold in practice [34]. In summary, case studies are an appropriate method to improve our understanding of BizDevOps teams and to show how relationships between planning and development and between operations processes of the SDLC are implemented.

Table 1. Primary and secondary data

Primary Data		
No. of Interviews	**Role of Interviewee**	**No. of Interviews per Role**
23 (some interviews were held with more than one interviewee)	CTO	1
	IT Manager	15
	Product Owner	3
	Team Member	9
	Sum	28
Over 400 pages of transcriptions	The interviewees were mainly conducted personally through face-to-face interviews. Some interviews were held via telephone. The research team took notes regarding observations during the interviews. In total, more than 400 pages of transcriptions and memos per interview were created between the end of 2016 and February 2018.	
Secondary Data		
Webpages, blog articles, and white papers	We searched through the internet and collected information about the companies. Often, the companies have blogs where they publish information about collaboration.	

We conducted an exploratory case study to answer our research question. Case studies offer a great variety of techniques for data collection [33] for the DevOps teams. Their characteristics and effects on the firms are primarily studied through expert interviews. An expert is someone who has privileged and deep knowledge about a special topic [35]. Here, the experts have privileged knowledge about DevOps teams and the customer view in their organization and can assess their characteristics and outcomes. We tried to talk to people in different roles and responsibilities to find out how planning processes and customer view (Biz) are implemented and how the DevOps concept fosters continuous innovation.

During our research, we used qualitative data-coding processes for the interpretation of our data [32, 36]. Furthermore, we followed the guidelines of grounded theory for data collection and analysis, as described by Wiesche et al. [15].

To participate in our study, a precondition was that the teams had to be familiar with the DevOps concept and must already have integrated planning, development, and operations processes. Additionally, the teams should have integrated an agile method (e.g. Scrum or Kanban) to collaborate with customers. We conducted a multiple-case study to analyze the relationships of business strategies with software development and operations. In short, we talked to 28 interview partners from 15 companies. Table 1 provides information about primary and secondary data.

A semi-structured interview was conducted with each participant, supported by guidelines and a list of questions or general topics that the interviewers wanted to touch upon [33]. The questions were mainly open-ended, giving the interviewees the possibility to explore their experience and views [33].

The interview guidelines helped to keep the interaction focused as data collection proceeded. It ensured comparability of data across individuals, settings, and researchers [37]. Although the interview process was systematic and comprehensive, the interviewer had a high degree of freedom to probe and explore these guidelines. Thus, questions were adjusted during the interviews to gain more in-depth knowledge for each case.

Each interview lasted about 45–75 minutes and was conducted through face-to-face meetings or by telephone. The interviews were held in German or English. German statements were translated into English for further analysis. Every interview was recorded and transcribed. Moreover, a lot of notes were taken during the interview.

Table 2. Coding process and core categories

Dimension	Definition	Statements (examples)
Team side	The responsible person for planning the backlog, integrated within the DevOps team.	*“From the developer's point of view, I'm very happy that the PO now sits next to me and I can have a constant exchange with him” (Team member).* *“We have a lot of cross-functional teams. That means you have a team with product manager, developers, and QA” (CTO).*
Customer side	The responsible person for planning, integrated at the customer/ business side	*“We work together relatively closely with PMs from other company parts, although they do not sit with us” (IT manager).* *„They are also [organizationally] close to us in the holding company. [...] they call themselves business class” (Team member).*
Team lead	The responsible person of the planning process.	*“I'm in the team in some roles, i.e. in this product team as Product Owner. At first, I also did a lot of development by myself, but everything else the teams do, they just have to vote with the customer” (PO/CTO).*

During our data analysis, we wanted to examine the relationships and concepts between business planning and software development [8]. The related literature presented in Chapter 2 was helpful for guiding our examination. For the coding process, we followed the guidelines approach presented by existing literature [15, 36] and used the software NVivo10. During the coding, the research team took notes to justify the coding process. Afterwards, we identified subcategories in the planning and collaboration processes with the help of axial coding. Finally, with the help of selective coding, we related the categories to mechanisms describing the effect of collaboration between BizDev and DevOps.

4. Findings

We started with an open coding process to identify core categories of relationships between BizDev and DevOps. The categories explain the process of planning in software development projects and different forms of customer integration into the SDLC. Table 2 presents our findings regarding the open-coding process. These findings confirm the results of existing research about integrating the customer view in software development projects with the help of a PO [38].

Table 3. Areas of planning relevant to DevOps teams

Area of Planning	Definition	Code Selection
Responsibility	Planning responsibility means that the team is now responsible and incorporated within the planning process and includes the impact of development and operations.	• Writing requirements • Service responsibility • Requirements implementation • Common understanding
Scope	Planning scope is defined as the size and extent of planned components that should be implemented by the team into the software in short iterations.	• Agile development meetings • Communication • Understanding of the software components and customer needs • Small increments
Dependency	Planning dependency is defined as the relationship of planning processes that are now integrated within the team with project success and running the software successfully.	• Collaboration between planning, building, and running • Team autonomy/flat hierarchies • Respect among team members • Consequences of failures • Shared tasks and understanding

4.1 Areas of Planning as Subcategories

We present different ways to integrate the customer perspective in the team. On examining the three possibilities of collaboration between business planning and software development and operations, we realized that planning is related to different processes in the SDLC and is connected to BizDev and DevOps activities.

Our findings confirm that the integration of a PO within a DevOps setting is important for achieving continuous innovation. The collected data show that apart from the establishment of the PO in the DevOps team, there are different areas of planning, as described in Table 3. Our interviewees stated that the responsibility for the tasks of planning, developing, and running the software is now integrated into one cross-functional IT project team.

"I know that the Product Owners, who came to us, were already relatively IT-savvy and partly from IT. They have taken a different career path and spent some time in marketing. […] That brought a lot of responsibility into the team" (Team member).

Furthermore, the scope of planning changed to a high degree. In traditional software development projects, the customers only have the possibility to plan their requirements for very long release cycles. Hence, business people cannot introduce a new demand into the development cycle because the planning phase is already closed and they have to wait a long time for adding new demands to the next big release [8]. This problem can be avoided through the implementation of continuous planning with the help of the BizDevOps approach, because introduction of new ideas and requirements is possible at all times.

"So, what distinguishes our team is mainly the Product Owner. A good planning and coordination with the Product Line Management specify the requirements. A really good planning with user stories and not just reacting to requirements but working proactively. […] We have always really attempted to show a minimal product finished in two weeks—a small increment that we were able to present" (IT manager).

Finally, we identified the area of planning dependencies as related to different processes in DevOps teams. Through the integration of a high degree of autonomy, the team is responsible for the planning process, as mentioned before, and must also be aware of the dependencies for the project success. Since the planning processes are now integrated in the team, the successful software delivery is in the hands of the BizDevOps team.

"Ultimately, the responsibility for the applications lies within the team, and that means—now that the Product Owner is also in the team—actually everything from writing the requirements to development to operation" (Team member).

We listed a row of areas of planning for BizDevOps teams. The introduction of these terms is necessary for describing the process of achieving continuous innovation in software development projects. These terms are dependent on the three core categories identified in Table 3. DevOps teams and business have already implemented planning configurations for achieving continuous innovations.

4.2 Mechanisms for Continuous Innovation

As mentioned before, a key prerequisite for continuous innovation is planning. Fitzgerald and Stol [6] state that innovation in business areas is connected to business value for the service recipient. Continuous innovation tries to enable processes that help to react to new market conditions and are related to metrics across the SDLC. Table 4 presents the mechanisms for meeting continuous innovations related to planning processes, identified with the help of our data. The following table depicts the mechanisms and the related area of planning with the key challenges that appear in our data.

For achieving continuous innovation, our cases initially implemented some mechanisms, which we identified as scalability, security, and quality. Innovation could be gained through scalable services. Scalability fosters speed to easily broaden the resources. The BizDevOps team is now able to do a lot of tasks by itself, because of responsibility for example, and hence to enhance scalability and speed of the service.

"Suddenly, we wanted to scale and that was not so easy in the old structure. With the scaling comes the fact that you want to bring things faster to the customers […] e.g., during Christmas time" (Team member).

The data presents insights that combine planning processes in DevOps teams; a higher level of unique selling points can be achieved through planning. The team is able to plan its demands and efforts to be taken in case of problems; thus, the team has a great overview.

"*Where we want to distinguish ourselves from competitors, which means where we have a higher level of competition there. We also want to have a higher level of agility, in other words, we specifically selected this DevOps-oriented approach because if we want the highest possible speed, then the team should be cross-functional"* (CTO).

However, to achieve scalability, a tight exchange of planned requirements is necessary. Furthermore, one interview partner mentioned that they still lost speed because the business department wanted a manual acceptance test: *"Hence, we have to wait for implementation, and I am angry about that"* (Team member).

We identified security to foster continuous innovations as the next mechanism. This is related to the responsibility and scope that are now integrated in the teams. Planning responsibility delivers a feeling of safety to the team; the team is no longer concerned that new requirements would be introduced by external people because they are involved in the planning processes.

"Teams have a higher flexibility; they are more autonomous. They have a higher degree of safety regarding planning" (IT manager).

Security is important during the entire SDLC, and some team leads still make great efforts to *"claim operations responsibility"* (IT manager), because possible failures in the running systems need planning efforts as well.

The third mechanism that we identified is quality. Our findings indicate that a stable running software, where changes are possible when necessary, fosters innovations. A high-quality planning process avoids failures in the implementation and running phase of the software. BizDevOps teams need the skills and awareness to deliver the complete SDLC.

"The team and its members have very a high level of personal maturity. They have high claims to themselves and to the quality of work. They have a very high degree of customer-oriented thinking" (IT manager).

However, for achieving high stability for a system, planning quality and dependent factors should be considered. *"We want the team to work self-responsibly and decide when things go live and they have to take over the complete responsibility for quality and operations"* (CTO).

5. Discussion

In our study, we present rich descriptions of the combination of planning areas and mechanisms for achieving continuous innovations through BizDevOps teams, as presented in Figure 2. We

Table 4. Mechanisms for continuous innovation and planning relation

Mechanisms	Definition	Manifestation
Scalability	BizDevOps teams want to achieve the highest agility and speed to stand out from competitors through integrated planning and solving problems during run time operations and scaling the software to a broader level if necessary.	• Planning of necessary proportional increase of service resources • Claim against competitors • Enhancement of speed through fast reactions and planning autonomy
Related to	Planning responsibility, scope, and dependencies	• Need for quicker communication • Avoiding of manual acceptance test by PO
Security	BizDevOps teams are responsible for planning and hence want to achieve high planning security, because the team defines the priority of the planned increments of the software if failures appear in the running software.	• Scope and autonomy lead to planning security • Responsibility for services and consideration in planning processes
Related to	Planning responsibility	• Need for communication within the team • Call for taking over complete service responsibility
Quality	BizDevOps teams are responsible for delivering and running the software. The team members need to be aware that they are responsible if problems appear. Proactive avoidance of failures is enhanced through a high-quality planning processes for the product.	• Product quality depends on planning, developing, and running tasks • Team members develop awareness of product quality
Related to	Planning scope and dependencies	• Need for accurate planning of the backlog • Call for awareness of failures

identified areas in BizDevOps and planning processes that trigger mechanisms for achieving innovations for customers, as shown in Tables 3 and 4.

We give insights into how the phenomenon of BizDevOps can be arranged with the related planning processes to understand how continuous innovation can be achieved. This is different from the findings of existing literature, where the need for further investigation of continuous planning and innovation and DevOps capabilities is highlighted [8, 19]. Thus, the grounded theory approach of achieving continuous innovation through DevOps teams developed in this paper is the first attempt to explain of how planning processes are integrated and how the BizDevOps concept can be achieved.

We propose that this continuous innovation is correlated with the planning of customer requirements. Furthermore, we find that the evolution of continuous innovation is a process that is combined with planning areas. There are some challenges described by our interviewees that need to be considered.

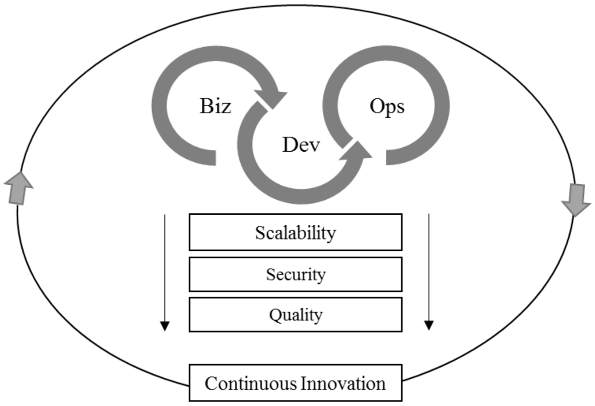


Figure 2. Rich descriptions of achieving continuous innovation through BizDevOps

The first mechanism is scalability. Scalability depends on the planning responsibility of BizDevOps teams. The team members have to feel responsible for the service. If the scaling of the service is necessary due to some peaks, it must be recognized proactively by the team members and accurately planned and realized. Hence, it involves planning responsibility and the corresponding steps. Furthermore, to achieve the necessary speed for scaling, planning dependencies are important as well. The awareness of collaboration and the dependencies of scaling for project success are important points that should be considered.

The second mechanism is security. Security is related to planning responsibility. The BizDevOps team is responsible for planning and operating the system. Hence, there is a need for accurate planning and the awareness must be fostered in the team.

The third mechanism is quality. Quality is dependent on planning scope and dependencies. Within BizDevOps teams, a new culture of collaboration is necessary. The team members need to have the attitude that the service is owned by them. The members have to decide which components are to be developed in each iteration. Therefore, high quality in the planning process is necessary, so that less failures appears.

Additionally, we identified three dimensions of integrating the customer view (Biz) into the DevOps team. Our data indicate that the PO is in the DevOps team, on the customer side, or the team lead. We found evidence that if the PO is integrated within the DevOps team, the best collaboration and planning processes are achieved. However, our findings also indicate that if the PO is settled on the business side, a high degree of exchange and close collaboration with the business are achieved. The third dimension is when the PO is the team lead. This setting appears because of the transformation of traditional IT functions to a cross-functional BizDevOps team setting. Our data highlight that middle management positions could break away, leaving the company with a *"social responsibility"* (Team lead) against managers and the PO position is handed over to them.

5.1 Limitations and Future Research

As mentioned in Chapter 3, we mainly talked to IT managers and other IT people. This is because some of our cases lack even a typical business department. Two cases mainly consist of IT departments and some support units. Further research should include cases that have a traditional business department which is involved in the planning processes. Generalization of this study is limited by a case-study approach; hence, validity is limited to our findings. Other studies in different settings might complement our examination. We present insights into how the planning process could be integrated in a DevOps teams with BizDevOps. We present mechanisms for continuous innovation and planning relation, but this needs further enhancements. One way to achieve some kind of evaluation is by conducting a quantitative study. Further research is needed for the replications of our findings across other settings [33]. Additionally, future examination is necessary to identify other continuous innovations mechanisms that may be important for BizDevOps settings. Furthermore, we focus on intra-organizational collaboration of customers and IT functions. Other settings with inter-organizational, e.g. outsourcing, settings for development and operations tasks may provide new insights as well.

6. Conclusion

Our paper presents insights into continuous innovation steps that could be provided by BizDevOps teams by integrating the planning of customer demands. This is one of the first studies to investigate how planning processes and the customers view can be integrated in a DevOps team—i.e. integrated business processes. With the help of grounded theory, we derived rich descriptions for achieving continuous innovations through the concept of BizDevOps. Thus, we contribute to existing research [8, 27] and provide deeper insights into how the collaboration of business and IT functions could be implemented through new approaches like DevOps teams. Based on our explorative case study, we identified scalability, security, and quality as mechanisms for continuous innovation. Scalability includes the planning processes in case of an increase in service resources, accentuation against competitors, and the enhancement of speed through the possibility of fast reactions. Security in BizDevOps teams refers to the scope and autonomy that leads to planning security and the responsibility for the service and consideration in planning processes. Quality means that the product is dependent on planning, developing, and running tasks that are conducted by the BizDevOps teams and that team members need to develop an awareness of product quality. For practice, we present guidelines for closer collaboration and integration of planning processes and short cycle times in software development projects.

7. Acknowledgement

We thank the German Federal Ministry of Education and Research (BMBF) for funding this research (grant code 03FH005PX4).

8. References

[1] Holmqvist, M., and Pessi, K., "Agility through Scenario Development and Continuous Implementation: A Global Aftermarket Logistics Case", European Journal of Information Systems, 15(2), 2006, pp. 146-158.

[2] Lowry, P.B., and Wilson, D., "Creating Agile Organizations through IT: The Influence of Internal IT Service Perceptions on IT Service Quality and IT Agility", The Journal of Strategic Information Systems, 25(3), 2016, pp. 211-226.

[3] Fitzgerald, B., and Stol, K.-J., "Continuous Software Engineering and Beyond: Trends and Challenges", in: International Workshop on Rapid Continuous Software Engineering, ACM, 2014.

[4] Wiedemann, A., and Wiesche, M., "Are You Ready for Devops? Required Skill Set for Devops Teams", European Conference on Information Systems, 2018.

[5] Markus, M.L., and Keil, M., "If We Build It, They Will Come: Designing Information Systems That People Want to Use", Sloan Management Review, 35(4), 1994, pp. 11-35.

[6] Recker, J., Rosemann, M., Indulska, M., and Green, P., "Business Process Modeling-a Comparative Analysis", Journal of the Association for Information Systems, 10(4), 2009, pp. 333-363.

[7] Wiesche, M., and Krcmar, H., "The Relationship of Personality Models and Development Tasks in Software Engineering", ACM SIGMIS-CPR, 2014, pp. 149-161.

[8] Fitzgerald, B., and Stol, K.-J., "Continuous Software Engineering: A Roadmap and Agenda", Journal of Systems and Software, 123, 2017, pp. 176–189.

[9] Lehtola, L., Kauppinen, M., Vähäniitty, J., and Komssi, M., "Linking Business and Requirements Engineering: Is Solution Planning a Missing Activity in Software Product Companies?", Requirements Engineering, 14(2), 2009, pp. 113-128.

[10] Chen, L., "Continuous Delivery: Overcoming Adoption Challenges", The Journal of Systems and Software, 128, 2017, pp. 72–86.

[11] Knight, S., Rabideau, G., Chien, S., Engelhardt, B., and Sherwood, R., "Casper: Space Exploration through Continuous Planning", IEEE Intelligent Systems, 16(5), 2001, pp. 70-75.

[12] Pflügler, C., Wiesche, M., and Krcmar, H., "Subgroups in Agile and Traditional IT Project Teams", Hawaii International Conference on System Sciences, 2018.

[13] Overby, E., Bharadwaj, A., and Sambamurthy, V., "Enterprise Agility and the Enabling Role of Information Technology", European Journal of Information Systems, 15(2), 2006, pp. 120-131.

[14] Silverthorne, V., "Business Process Management Aids Application Development", in: Application Development Handbook, TechTarget, E-Handbook, 2017.

[15] Wiesche, M., Jurisch, M.C., Yetton, P.W., and Krcmar, H., "Grounded Theory Methodology in Information Systems Research", MIS Quarterly, 41(3), 2017, pp. 685-701.
[16] Peppard, J., "Rethinking the Concept of the IS Organization", Information Systems Journal, 28(1), 2018, pp. 76-103.
[17] Horlach, B., Drews, P., Schirmer, I., and Boehmann, T., "Increasing the Agility of IT Delivery: Five Types of Bimodal IT Organization", Hawaii International Conference on System Sciences, 2017.
[18] Wiedemann, A., and Wiesche, M., "How to Implement Clan Control in DevOps Teams", Americas Conference on Information Systems, 2018
[19] Sebastian, I.M., Ross, J.W., Beath, C., Mocker, M., Moloney, K.G., and Fonstad, N.O., "How Big Old Companies Navigate Digital Transformation", MISQ Executive, 16(3), 2017, pp. 197-213.
[20] Debois, P.A., "Opening Statement", Cutter IT Journal, 24(8), 2011, pp. 3-5.
[21] Przybilla, L., Wiesche, M., and Krcmar, H., "The Influence of Agile Practices on Performance in Software Engineering Teams: A Subgroup Perspective", ACM SIGMIS-CPR, 2018.
[22] Lassak, S., Przybilla, L., Wiesche, M., and Krcmar, H., "Explaining How Agile Software Development Practices Moderate the Negative Effects of Faultlines in Teams", Australasian Conference on Information Systems, 2017.
[23] Sambamurthy, V., Bharadwaj, A., and Grover, V., "Shaping Agility through Digital Options: Reconceptualizing the Role of Information Technology in Contemporary Firms", MIS Quarterly, 27(2), 2003, pp. 237-263.
[24] Tripp, J.F., Riemenschneider, C., and Thatcher, J.B., "Job Satisfaction in Agile Development Teams: Agile Development as Work Redesign", Journal of the Association for Information Systems, 17(4), 2016, pp. 267 – 307.
[25] Châlons, C., and Dufft, N., "The Role of IT as an Enabler of Digital Transformation", in: The Drivers of Digital Transformation, Springer, Saarbrücken, Germany 2017, pp. 13-22.
[26] Brown, C.V., and Ross, J.W., "The Information Systems Balancing Act: Building Partnerships and Infrastructure", Information Technology & People, 9(1), 1996, pp. 49-62.
[27] Ross, J.W., Sebastian, I., Beath, C., Mocker, M., Moloney, K., and Fonstad, N., "Designing and Executing Digital Strategies", International Conference on Information Systems, 2016.
[28] Lwakatare, L.E., Karvonen, T., Sauvola, T., Kuvaja, P., Olsson, H.H., Bosch, J., and Oivo, M., "Towards DevOps in the Embedded Systems Domain: Why Is It So Hard?", Hawaii International Conference on System Sciences 2016.
[29] Baskerville, R., and Pries-Heje, J., "Short Cycle Time Systems Development", Information Systems Journal, 14(3), 2004, pp. 237-264.
[30] Christopher, M., and Towill, D.R., "Supply Chain Migration from Lean and Functional to Agile and Customised", Supply Chain Management: An International Journal, 5(4), 2000, pp. 206-213.
[31] Ries, E., The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses, Crown Business, New York 2011.
[32] Urquhart, C., Lehmann, H., and Myers, M.D., "Putting the ‘Theory’ Back into Grounded Theory: Guidelines for Grounded Theory Studies in Information Systems", Information Systems Journal, 20(4), 2010, pp. 357-381.
[33] Yin, R.K., Case Study Research: Design and Methods, Sage Publications, Thousand Oaks, CA, 2014.
[34] Flyvbjerg, B., "Five Misunderstandings About Case-Study Research", Qualitative Inquiry, 12(2), 2006, pp. 219-245.
[35] Bogner, A., Littig, B., and Menz, W., "Introduction: Expert Interviews—an Introduction to a New Methodological Debate", in: Interviewing Experts, Springer, 2009, pp. 1-13.
[36] Glaser, B.G., and Strauss, A.L., Grounded Theory: Strategien Qualitativer Forschung, Huber, Bern, 1998.
[37] Maxwell, J.A., Qualitative Research Design: An Interactive Approach, Sage Publications, Thousand Oaks, 2012.
[38] Pries-Heje, L., and Pries-Heje, J., "Agile & Distributed Project Management: A Case Study Revealing Why Scrum Is Useful", European Conference on Information Systems, 2011.

How to Implement Clan Control in DevOps Teams

Completed Research

Anna Wiedemann
Neu-Ulm University of Applied Sciences, Germany
anna.wiedemann@hs-neu-ulm.de

Manuel Wiesche
Technische Universität München, Germany
wiesche@in.tum.de

Abstract

Clan control is an important factor for project success. A clan is a group with strong social capital. Hence, the implementation of social capital is a precondition for clan control and project success. However, for managers it is still a challenge to build and motivate clan control within IT projects, because people have different skills and backgrounds. Especially in rapid application development projects when new project methods or approaches like agile software development or DevOps are applied. Thus, we build up on existing research and explore how clan control can be build and leverage within cross-functional DevOps teams. With the help of expert interviews with team leads and managers of different DevOps teams, we derived structural, cognitive and relational ties of social capital. Additionally, we present leverage factors from our data for social capital to facilitate clan control. Finally, we give implications how these factors can be implemented.

Keywords

DevOps, IT project management, Clan control, Social capital

Introduction

More and more organizations organize work in team based structures (Kirsch et al. 2010). Reasons therefore are, that work gets increasingly complex, knowledge intensive and depends on non-routines. With the help of teams, organizations want to react to changing situations (Goh et al. 2013). Teams compose of individual people with different knowledge as well as skills and can act in a collaborative manner to realize a specific aim (Kirsch et al. 2010; Klendauer et al. 2012; Towry 2003; Wiesche and Krcmar 2014). IT project success is still an aim that is difficult to achieve.

One possibility to achieve collaboration is control, it *"is defined as any attempt to align individual behaviors with organizational objectives"* (Wiener et al. 2016, p. 742). To achieve a desired behavior, formal as well as informal control can be used (Heumann et al. 2015; Kirsch 2004; Ouchi 1978). In context of teams, it is difficult to rely only on formal controls because of individual behaviors and the complexity to measure personal contributions to team success (Kirsch et al. 2010; Schermann et al. 2012; Towry 2003). Information systems (IS) project control is essential to drive or adjust project stakeholders' behavior to motivate them to achieve project aims (Kirsch et al. 2010; Wiener et al. 2016). The literature highlighted that there is a relationship between social capital and informal clan control (Chua et al. 2012; Kirsch et al. 2010; Liu et al. 2015; Wiener et al. 2016).

Clan control is defined as a type of informal control. It appears when the team members behavior is motivated by shared norms, values and strong affiliation feelings to the group with a common goal (Kirsch et al. 2010). Clan control is a phenomenon that occurs on team level. Teams' social factors play an important role. Moreover, the literature highlighted that social factors are an essential prerequisite of clan control and next to team members, managers play an important role to generate clan control (Kirsch et al. 2010; Towry 2003). Mangers are often part of the team when they work within or very close with the team (Kirsch et al. 2002; Liu et al. 2015).

In the recent years, the use of agile methods in IT projects gained wide acceptance. It has become a major driver for performance within a lot of IT functions (Cram and Newell 2016; West et al. 2010). Software development projects remain a major concern for IT managers. Today, companies like Kaiser Permanente apply an approach called DevOps that goes beyond agile (Ross et al. 2016), it is compound of "development" and "operations". This approach helps IT departments to transform to service-centric IT operating models. With the help of DevOps, IT departments are enabled to reduce their software cycle times and bring new features into production in very short time (Ross et al. 2016; Sebastian et al. 2017).

The literature highlight that implementing DevOps capabilities will become a competitive requirement for incumbent companies (Sebastian et al. 2017). Managers are challenged to bring team members closer together for achieving project success (Chua et al. 2012; Majchrzak et al. 2005). Prior research depicts that social capital could be helpful for enhancing team work (Liu et al. 2015). Social capital is defined as "*the sum of the actual and potential resources embedded within, available through, and derived from the network of relationships possessed by an individual or social unit. Social capital thus comprises both the network and the assets that may be mobilized through that network"* (Nahapiet and Ghoshal 1998, p. 243). Since project members often have different backgrounds from several professions that need to achieve culture of collaboration (Chua et al. 2012; Wiesche and Krcmar 2014). Within DevOps teams barriers between development and operations people could appear. There is a need to solved these barriers to gain effective collaboration (Fitzgerald and Stol 2017). Often complex tasks need to ab addressed by the team, therefore strong collaboration with team members and shared values, understanding as well as social cohesions are necessary (Liu et al. 2015). Hence, we put forth the following research questions: *How can managers build and leverage social capital within DevOps teams to achieve high clan control?*

We argue that managers can promote clan control within team settings for example team-based clan control through the usage of social capital and the implementation of leverage factors. We derived by these factors with the help of a qualitative investigation IT managers that are responsible for DevOps teams. We investigate how social capital influence the collaboration within DevOps teams. Furthermore, we provide three factors that influence the relationship between social capital and clan control.

Related Literature

DevOps Teams

Since the presentation of the agile manifesto in 2001, agile software development methods gained high popularity (Tripp et al. 2016; West et al. 2010). A prerequisite of agile development teams is the ability to react to new and unforeseen situations, for example new customer demands, with the help of a team (Wiedemann and Weeger 2017). The aim is to handle the complexity of fast changing environments and to achieve better software quality as well as higher customer satisfaction. Thus, more and more IT functions are implementing product-oriented agile IT teams (Burke et al. 2006; Lassak et al. 2017; Pflügler et al. 2018; Przybilla et al. 2018).

The literature shows that research on agile methods mainly focuses on development activities (Goh et al. 2013; Tripp et al. 2016). IT functions have to enable a tighter collaboration between the different units of development and operations of an IT function to ensure that errors are fast fixed. Hence, the quality and resilience of the software is enhanced. The literature presents that IT functions' activities of development and operations have to be continuous (Fitzgerald and Stol 2017). DevOps helps to combine these approaches. For fast providing of new software features and reacting on problems, IT departments should implement cross-functional teams rather than separated silo IT departments (Fitzgerald and Stol 2014).

Within many IT organization customers gain deployments of new software code very seldom (Lwakatare et al. 2016). One reason is the poor exchange in form of communication and collaboration between the two department of software development and operations (Fitzgerald and Stol 2017). To address this issues, DevOps provides solutions to avoid interruptions between different stages of the software delivery process (Fitzgerald and Stol 2014). The software development life-cycle involves planning, building and, running processes. DevOps helps companies to implement speed and flexibility to deliver rapid development and implement digital innovations when they are needed (Ross et al. 2016). Hence, possible risks through software releases can be reduced, and feedback on a new software deployment can be faster received (Lwakatare et al. 2016). To summarize, the aim of the DevOps concept is to foster collaboration, automation, virtualization as well as implementing tools for bringing activities of software development and operation closer together (Humble and Molesky 2011; Lwakatare et al. 2016).

Social Capital and Clan Control

Social capital theory is important for collaboration and addresses the social and cultural preconditions (Riemer and Klein 2008). The literature presents that managers can support and influence a project by social capital. Two forms of control exist, namely formal and informal (Kirsch et al. 2010; Ouchi 1979). Formal control focuses on hierarchical power and authority that influence team members to act in a specific way. Informal control relies on self-control through individuals characteristics and/or social relationships (clan control) (Kirsch et al. 2010; Liu et al. 2015). Prior research presents that there are relationships between social capital and clan control (Chua et al. 2012; Kirsch et al. 2010). A clan is defined as a homogenous group with individuals that share common beliefs, values and norms (Liu et al. 2015; Ouchi 1979).

Social capital can be measured with the help of three dimensions namely: structural, cognitive, and relational dimensions. These dimensions can be implemented as interrelated ties between team members (Nahapiet and Ghoshal 1998; Wagner et al. 2014). The structural dimension of social capital comprises the contact between the individuals of the network (Nahapiet and Ghoshal 1998). Structural ties include the settings and forms of communication meetings of the project members which are typically described by frequency, centrality, stability, and density. Communication exchange can happen physically by colocation, or virtually by emails (Liu et al. 2015; Wagner et al. 2014). The cognitive dimension is defined as the assets that deliver shared understanding and importance among the stakeholders (Nahapiet and Ghoshal 1998). This dimension includes common language, shared codes, narratives and perspectives as well as the project stakeholder's interpretation of reality (Wagner et al. 2014). The relational dimension views on the specific relationships that the people of the relationship have (Nahapiet and Ghoshal 1998) and includes trust and respect which is a precondition for knowledge sharing. Furthermore, the stakeholders see each other as partners and consult each other for better working together (Wagner et al. 2014).

Social capital is related with networks and relationships between individuals which enhance collaboration. Prior research treats groups with strong social capital and clans as interchangeable (Chua et al. 2012). Managers can be a part of clans if they work close together with the members (Kirsch et al. 2002). Clan control can be established by the building of social capital. Managers should guide social capital in the clan to reinforce shared values, beliefs and norms that are helpful for project success (Chua et al. 2012). Enacting clan control is bilateral, by building and leveraging the clan. In the first step social capital is developed by structural, cognitive, and relational ties and in the second step, social capital can be leveraged for a specific outcome (Chua et al. 2012; Liu et al. 2015). Within this paper, we derived factors for social capital, afterwards we present how these factors can be leveraged. This research focuses on clan control, because prior research depicts managers could be team members too. We are interested in building relationships between team and manager (Liu et al. 2015). Hence, we are examine the development of for example shared norms, values and do not focus on self-control characteristics like self-monitoring of behavior (Wiener et al. 2016).

Research Methodology

To answer our research question, we decide to conduct an exploratory research. By using a qualitative research approach, our research can be characterized as interpretive. We want to understand how social capital influences DevOps teams and how they are implemented within existing IT function. Hence, we conduct a row of expert interviews with managers and team leads of DevOps teams. Due to the fact that DevOps is a very unexplored research topic, it calls for case study design (Yin 2009). The case study approach is defined as *"an empirical inquiry that investigates a contemporary phenomenon in depth and within its real-life context"* (Yin 2009, p. 18). Case studies are a suitable approach in rare areas in which researcher as well as practitioners search for new insights (Cao et al. 2013; Eisenhardt 1989). Thus, new findings and a deeper understanding of the topic can be achieved. DevOps settings have to be investigated in their daily environment to gain insights about that approach and how the teams are working with that concept. Hence, qualitative methods are appropriate if the existing body of knowledge lacks of information and questions are open, which need further investigations.

Site Selection

For the investigation, the focus laid on DevOps teams. Therefore, we identified contact persons, which are engage to the DevOps concept. The focus laid on people, who have experience with DevOps and management position within a IT organization. For the expert interviews, over 50 companies from different industries were contacted via e-mail and telephone, the premise for study participation was that they have already implemented a DevOps team. Our aim was to find cases that have an interesting setting or are easy to replicate to achieve similarities and variation about our findings (Eisenhardt 1989). In each company, minimum one manager (e.g. team lead) was interviewed regarding their control mechanisms of DevOps teams. The aim was to gain the leadership view on how social capital influence the social relationship with DevOps teams. In summary, we talked to ten managers from seven companies with help of eight interviews (interviews in company 1 and 2 were conducted with both managers at the same time).

ID	Role	Team	Work Experience	Industry	DevOps since
I1	Manager	Team 1	10 -15 years	Media	six month
I2	Manager		10-15 years		
I3	CTO	Team 2	5-10 years	Furniture	two years
I4	Manager		5-10 years		
I5	Manager	Team 3	15-20 years	Fashion	three years
I6	Manager	Team 4	10-15 years	Service	five years
I7	Manager	Team 5	10-15 years	Energy	> five years
I8	Manager	Team 6	10-15 years	Health Care	three years
I9	Manager		15-20 years		
I10	CTO/Manager	Team 7	10-15 years	Media	two to three years

Table 1. Characteristics of the Interview Partners

Table 1 depicts that we interviewed managers from different industries. The interviewees had different positions for example CTO or manager. Managers are members of the team and/or responsible for one or more teams. The cases have experience with the DevOps approach between six month and five years.

Data Collection and Analysis

The data collection phase took place from October 2016 through November 2017. After identifying possible interviewees, semi-structured interviews were conducted with the participants. Each interview had a duration about 45-75 minutes and was carried out primarily through face-to-face meetings or by telephone. In exploratory research personal interviews are recommended because they allow comprehensive discussions. The interviews were held in German or English language. German statements were translated into English for further analysis. The questions were mainly open-ended, that the interviewee had the possibility to explore their experiences and views (Yin 2009). The questionnaire comprises questions regarding general background of the organization, the use of DevOps principles, and the implementation and enhancement of social capital (e.g. how do the stakeholders communicate?). The interviews delivered insights into the working style of the teams and informal control mechanisms of the DevOps team. Every interview was recorded.

With the help of the expert interviews with the IT managers, we were able to derive general constructs for social capital as well as factors that leverage clan control. We used coding processes to investigate the relationships between managers and DevOps teams. Coding is very helpful to identify categories and corresponding sub-categories and for examining relationships (Runyan et al. 2007; Sarker et al. 2013). The interview data was coded using the software NVivo 10. We started the coding process following the guidelines of Miles and Huberman (1994). Hence, we started with an open coding process. During the coding, the research team took notes to justify the coding section. Afterwards, the results were analyzed regarding the presented ties of social capital. Then the research team compared their findings for each dimension to identify commonalities, relationships and patterns. The focus laid on the constructs which we identified from literature and the new capabilities that emerged during the data analysis. Furthermore, we analyzed the findings regarding relationships to identify the degree leverage factors within the teams.

Findings

We analyzed the cases to identify which forms of social capital are established to enhance relationships through the managers and DevOps teams. We focus on social capital that is built by the managers in each DevOps team to generate ties (Liu et al. 2015; Nahapiet and Ghoshal 1998). Table 2 provides insights how social capital can be built by managers and are helpful to build effective clan control (Chua et al. 2012).

Construct	Constructs	Example Quotes
Structural ties	Enable physical and virtual meetings that support the group.	*“We have eight people in India and two in Germany, […] they work very close together on a cross-border and daily basis. At the moment we have an overlap of four hours that is very good.”* I8
	Integrate autonomous teams with product responsibility.	*“We are in the process of integrating completely self-organized teams, which means that they are organizationally and technically independent.”* I5
	Implement different forms of meeting structure (team intern and extern).	*“Lectures outside the box so that you look at new tools, new frameworks, new developments.”* I7
Cognitive ties	Spread knowledge within the team and company.	*“We do not have pockets of knowledge. Of course we try to avoid that very much. So it is the worst case, if something like that happens.”* I6
	Establish cross-functional teams with shared knowledge.	*“We do not always have everything to do top down by default we can look how other teams do that.”* I2
Relational ties	Generate awareness of product ownership.	*“That's the part that I'm proud of - we noticed that the teams are going to grow beyond themselves and also bear extra effort to deliver the service in a high quality.”* I4
	Enable freedom within the team.	*“It is nice to see how this freedom develops the team positively.”* I1
	Give responsibility and trust into the team.	*“Management positions will break away and there is a social responsibility towards managers […] that are 15 years or longer within the company.”* I2

Table 2. Social Capital used in DevOps Teams

Table 2 presents activities that helps to build a clan but our findings present that distinct actions of managers are necessary to motivate the team after building social ties. This is in line with the findings of Chua et al. (2012) there are two ways of managing a clan proactively. The first one is to reinforce shared norms, values and beliefs that are important for projects and the second one is to constrain beliefs, norms and values that endanger project success. Table 3 presents insights of the purpose and necessary steps that were used by the managers to motivate the team to achieve high clan control. The order presents the frequency and importance of the constructs, for example call for operations responsibility was a major concerns of our informants.

Leverage factors	Purpose for managers	Necessary Steps
Call for operations responsibility	Team members need to overtake responsibility for operations.	Build the **awareness** for the complete software delivery life-cycle. Team members are equally responsible for development and operations tasks. But the fear of being overstrained should be avoided. If the system is not working, the

		team is responsible not any another unit in the IT department.
Integrate a culture of feedback and learning	Team members must work within feedback culture and enable continuous learning.	Foster the **willingness** to give constructive feedback and move from a finger-pointing culture to a feedback and learning culture.
Establish receptiveness for fast changes	Team members need to react as soon as possible to changing demands or failures.	Enhance the **attitude** for continuous delivery in fast changing environment. Team members have to learn new actions from development and operations as well as stay up to date to handle new complex changes.

Table 3. Leverage Factors, Purpose and Necessary Steps for Managers

Table 3 depicts the norms and values that could be fostered by managers to leverage clan control. Despite a high level of social capital, these factors could help DevOps teams to achieve a high level of clan control.

Discussion

Our study investigates the implementation of social capital within cross-functional teams. We examined how managers can foster social capital within DevOps teams and what are major challenges for achieving clan control. Now, we want to discuss the major challenges for building and leveraging social capital. Our findings suggest that high clan control is an important factor, because in DevOps settings, team leads move toward a team member position. Nevertheless, our results present that there is still a need for management and guidance of the DevOps team. In the following sections we explain how these relationships can be established.

Our findings indicate that managers have to make great efforts to implement new activities within the teams. For example, integrating operations activities into a development team is a great challenge. Prior research depicts that software operations and development differs in many ways (Edberg et al. 2012). For example developers wanted to implement fast new software features, but operations people want less changes in the system to guarantee stability (Edberg et al. 2012; Shaft and Vessey 2006). Hence, development and operations are usually controlled with the help of different modes. These tensions could complicate the building of a clan within projects. We present constructs of social capital and clan control to explain the approximation of different backgrounds that could be achieved through DevOps.

Furthermore, our study confirmed existing research that highlights that managers can be part of the clan (Liu et al. 2015). If the managers have a high business orientation, the product owner part of an DevOps team might be suitable position for them. Product owners are usually responsible for changes, refine and prioritized the product backlog, which is a list with new tasks that should be built for the system (Schlauderer et al. 2015). I10 demonstrate that he has overtaken the product owner role additionally to his CTO position and the *"team together with the product owner overtakes all necessary roles"* for managing the service.

Despite being a clan member, for integrating clan control in DevOps settings a suitable management style is necessary. Our findings give insights that the manager role is changing in this flat hierarchies. That means that the manager is seen as leadership person as well as a team member. Team lead I6 mentioned that *"there are very few situations where I really have to lead and control the team"* only in situations of for example major incidents the team leads *"phone is ringing first."* Additionally, one manager highlighted that *"an authoritarian leadership style in the development team is absolutely not working. Laisser-faire always has a bit of an approach that it is not effective. What actually comes out of my experience is an intrinsic motivation"* I7. The literature presents that control is necessary over several hierarchical levels and that cross-level difference exists in the control styles. Further, there exist different forms of control styles, coercive or enabling (Heumann et al. 2015). Hence, we studied different control mechanisms that are helpful to find out how the control style could be influenced. We highlighted that both forms of control styles are necessary. For example our findings and leverage factors present that for managing problems, clear guidelines need to be defined, and for integrating culture alignment, enabling styles are suggested. Hence, we summarize that a suitable control style is essential in DevOps teams that fosters intrinsic motivation, create awareness, willingness and attitude for the DevOps approach. We recommend further research in the area of which control styles are suitable for different control situations.

The literature gives insights in how to include social capital into virtual organizations to gain credibility (Riemer and Klein 2008). Our findings indicate that there is a necessity of structural ties too. Some managers highlighted that they have to organize virtual meetings because they work in offshore (team 6) and nearshore (team 1) team settings. To foster social capital, the interview partners mentioned that they organize regular personal meetings with the near- and offshoring colleagues to share knowledge, communicate and *"there is almost every week either someone from there here or someone from here there. I was there last week and that's very, very important. Especially with new concepts it is really much more effective to spend a week together"* I6. Hence, we summarize that virtual meetings are helpful for the management of existing and stable projects, but if new concepts like DevOps are used it is helpful to meet each team members personally.

Within the area of cognitive ties our findings present that a common and shared understanding as well as knowledge about the tasks of the software delivery life-cycle are indispensable. CTO I3 mentioned that the *"team is responsible from planning, operations, development, tests, and later for monitoring. The tasks are completely shared in the team."* A high degree of autonomy regarding decision making is typical in agile development projects (Cao et al. 2009). But within DevOps the team has to make decisions about their service that could have influence to the rest of the company. Hence the team must be able to communicate with other teams and managers if decisions *"concerns the system landscape"* I3.

Regarding relational ties, our findings confirm insights from prior research that trust is a major factor for successful team collaboration (Liu et al. 2015). Managers have to trust the team if problems appear *"the team tries to solve it, [if they cannot] an escalation processes will be started [...] this could not be achieved with a command and control leadership style"* I4. The *"ownership for the service"* I4 is changing because the complete team is responsible for the service and hence, a feeling of responsibility within in the team can be established.

For a high motivated and aligned team leverage factors are necessary (Chua et al. 2012). In our research we identified three leverage factors that are used by managers to achieve high level clan control: call for operations responsibility; integrate a culture of feedback and learning; establish receptiveness for fast changes. These factors should be considered by the managers when a DevOps team is already established. Managers should have actions implemented to build awareness *"the learning curve should be not too high and [new tasks] should be slowly acquired"* I6. Implementing DevOps approaches need a balance for the call for new responsibility and creating awareness for this necessity. The call for operations is an important leverage factor for managers. DevOps describes the development and operations tasks of the software delivery lifecycle, the team members have to overtake new responsibility. I1 mentioned that great efforts are necessary to *"claim for the operations responsibility"* of the team members, because a lot of team members are not willing to overtake this new responsibility.

Prior research shows that tight project schedules are often insufficient for building a clan (Chua et al. 2012). However, our findings present that DevOps teams are often organized in so-called product teams rather than in project teams. That means that they have no clear start or end date and the team members have to work there for an undefined time. Hence, we derive that clan control can be fostered in DevOps settings, because the ongoing team structure presents freedom and time for developing a clan. Additionally, in DevOps team settings it is important to deliver fast feedback to the team because there is usually a high degree of automation (Fitzgerald and Stol 2017). In project 1 *"these feedback loops were introduced, because we really put the operations pain into the development team and that first hurts and then it gets better"* I2. The team members need to know if something went wrong. Within DevOps settings, the team members should *"work in an open feedback culture and [...] always try to make decisions by consensus"* I10 and work on a common solution. To summarize, the integration of a culture of learning and feedback seemed to be a key factor for leveraging clan control. Furthermore, the receptiveness for fast changes is necessary since team members need to be motivated to work in a DevOps team. Team members need the attitude to steadily work on new solutions and provide it to production in short cycle times (Fitzgerald and Stol 2017). *"The most important thing is that people fit [in our team]"* I10 from a personal perspective. Therefore, different measures are possible for example the integration of *"an agile coach for the organization"* I3, that helps teams to learn the DevOps principles and distribute knowledge within the company.

Figure 1 present an overview of the interplay between social capital and leverage factors for clan control.

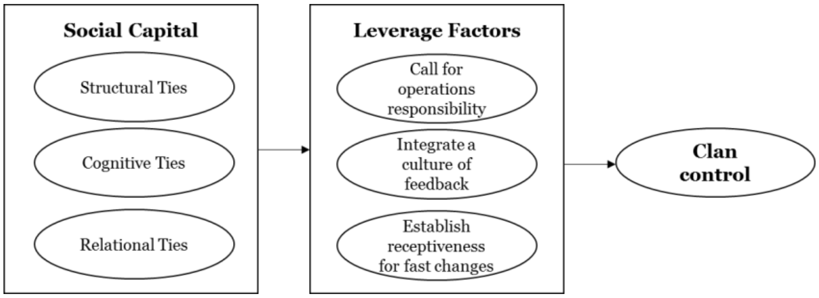

Social Capital
Structural Ties
Cognitive Ties
Relational Ties
Leverage Factors
Call for operations responsibility
Integrate a culture of feedback
Establish receptiveness for fast changes
Clan control

Figure 1. Clan Control as Process of Social Capital and Leverage Factors

Implications for Research and Practice

Our findings present insights into how managers can influence the build and motivation of clan control within DevOps teams. The present study enforces the necessity of informal control mechanisms in IT teams (Chua et al. 2012; Liu et al. 2015) and depict how social capabilities can be established and leveraged within DevOps teams. We contribute to existing research and enhanced prior findings of Chua et al. (2012) and Liu et al. (2015). We follow their call for further research in different project characteristics of different organizations and present an investigation of DevOps teams. Management of DevOps teams have an influence to motivate clan control as a controller to build structural, cognitive and relational dimensions of social capital. We deliver observations of social capital that could be facilitated by the managers within DevOps team settings and present a basis for further research. We enhance existing research on clan control in cross-functional IT teams and present DevOps specific social capital and three leverage factors that help to achieve clan control within these teams. Through the present paper an instigation of managing teams regarding their activities of the software delivery lifecycle with focus on software development as well as operations tasks is provided.

For practice we present guidelines for managing DevOps teams from an informal control perspective. We describe the changing role of a management positions to a more coaching and team member oriented role. Additionally, we identified challenges from our data and subsequently actions for managers how they can handle and motivate clan control. We derived three factors that leverage clan control if they are combined with the ties of social capital.

Limitations

The study provides insights into the clan control of DevOps teams, but has some limitations that need to be considered when interpreting the results. The generalizability of the findings is limited, because we only investigated one team and one or two manager perspectives in seven companies, this means that we only examined one DevOps team per organization. In addition, all the organizations are located in Germany. This entails that the applications of the results are limited. Further research could repeat the study in different countries, to examine more teams and different settings per company. Furthermore, a quantitative research could achieve validation for our findings.

Conclusion

In this paper we demonstrate how mangers can support collaboration within IT teams and build social relationship. With the help of DevOps teams, we examined the influence of social capital for collaboration. Managers should foster structural, cognitive and relational ties through concrete actions. We derived constructs of our data from expert interviews and depict the importance of social capital within project teams. For every tie of social capital, we delivered several constructs that provide insights on how social capital can be built with help of managers to support an IT team. Furthermore, we derived leverage factors for achieving a higher clan control if they are combined with social capital of the team. The three factors are: call for operations responsibility, integrate a culture of feedback and learning as well as establish receptiveness for fast changes. If managers could implement these factors and achieve awareness, willingness, and attitude for the DevOps approach, clan control could be leveraged.

Additionally, we described the changing role of managers within DevOps settings. Managers have to lead the teams on the one hand and act as team members on the other hand. These findings present possibilities for further research for example key characteristics of the leadership style for cross-functional DevOps.

Acknowledgment

We thank the German Federal Ministry of Education and Research (BMBF) for funding this research (grant code 03FH005PX4). The responsibility for the content lies with the authors.

REFERENCES

Burke, C.S., Stagl, K.C., Salas, E., Pierce, L., and Kendall, D. 2006. “Understanding Team Adaptation: A Conceptual Analysis and Model,” *Journal of Applied Psychology* (91:6), pp. 1189-1207.

Cao, L., Mohan, K., Ramesh, B., and Sarkar, S. 2013. “Adapting Funding Processes for Agile IT Projects: An Empirical Investigation,” *European Journal of Information Systems* (22:2), pp. 191-205.

Cao, L., Mohan, K., Xu, P., and Ramesh, B. 2009. “A Framework for Adapting Agile Development Methodologies,” *European Journal of Information Systems* (18:4), pp. 332-343.

Chua, C.E.H., Lim, W.-K., Soh, C., and Sia, S.K. 2012. “Enacting Clan Control in Complex IT Projects: A Social Capital Perspective,” *MIS Quarterly*, pp. 577-600.

Cram, A.W., and Newell, S. 2016. “Mindful Revolution or Mindless Trend? Examining Agile Development as a Management Fashion,” *European Journal of Information Systems* (25:2), pp. 154-169.

Edberg, D., Ivanova, P., and Kuechler, W. 2012. “Methodology Mashups: An Exploration of Processes Used to Maintain Software,” *Journal of Management Information Systems* (28:4), pp. 271-304.

Eisenhardt, K.M. 1989. “Building Theories from Case Study Research,” *Academy of Management Review* (14:4), pp. 532-550.

Fitzgerald, B., and Stol, K.-J. 2014. “Continuous Software Engineering and Beyond: Trends and Challenges,” in: *International Workshop on Rapid Continuous Software Engineering*. ACM, pp. 1-9.

Fitzgerald, B., and Stol, K.-J. 2017. “Continuous Software Engineering: A Roadmap and Agenda,” *Journal of Systems and Software* (123), pp. 176–189.

Goh, J.C.-L., Pan, S.L., and Zuo, M. 2013. “Developing the Agile IS Development Practices in Large-Scale IT Projects: The Trust-Mediated Organizational Controls and IT Project Team Capabilities Perspectives,” *Journal of the Association for Information Systems* (14:12), pp. 722-756.

Heumann, J., Wiener, M., Remus, U., and Mähring, M. 2015. “To Coerce or to Enable? Exercising Formal Control in a Large Information Systems Project,” *Journal of Information Technology* (30:4), pp. 337-351.

Humble, J., and Molesky, J. 2011. “Why Enterprises Must Adopt DevOps to Enable Continuous Delivery,” *Cutter IT Journal* (24:8).

Kirsch, L.J. 2004. “Deploying Common Systems Globally: The Dynamics of Control,” *Information Systems Research* (15:4), pp. 374-395.

Kirsch, L.J., Ko, D.-G., and Haney, M.H. 2010. “Investigating the Antecedents of Team-Based Clan Control: Adding Social Capital as a Predictor,” *Organization Science* (21:2), pp. 469-489.

Kirsch, L.J., Sambamurthy, V., Ko, D.-G., and Purvis, R.L. 2002. “Controlling Information Systems Development Projects: The View from the Client,” *Management Science* (48:4), pp. 484-498.

Klendauer, R., Berkovich, M., Gelvin, R., Leimeister, J.M., and Krcmar, H. 2012. “Towards a Competency Model for Requirements Analysts,” *Information Systems Journal* (22:6), pp. 475-503.

Lassak, S., Przybilla, L., Wiesche, M., and Krcmar, H. 2017. “Explaining How Agile Software Development Practices Moderate the Negative Effects of Faultlines in Teams,” in: *Australasian Conference on Information Systems*. Hobart, Australia.

Liu, G.H., Wang, E., and Chua, C.E.H. 2015. “Leveraging Social Capital to Obtain Top Management Support in Complex, Cross-Functional IT Projects,” *Journal of the Association for Information Systems* (16:8), pp. 707-737.

Lwakatare, L.E., Karvonen, T., Sauvola, T., Kuvaja, P., Olsson, H.H., Bosch, J., and Oivo, M. 2016. “Towards Devops in the Embedded Systems Domain: Why Is It So Hard?,” in: *Hawaii International Conference on System Sciences* Kauai, USA.

Majchrzak, A., Malhotra, A., and John, R. 2005. "Perceived Individual Collaboration Know-How Development through Information Technology–Enabled Contextualization: Evidence from Distributed Teams," *Information Systems Research* (16:1), pp. 9-27.

Miles, M.B., and Huberman, A.M. 1994. *Qualitative Data Analysis: An Expanded Sourcebook*. Thousands Oaks, Carlifornia: Sage Publications.

Nahapiet, J., and Ghoshal, S. 1998. "Social Capital, Intellectual Capital, and the Organizational Advantage," *Academy of Management Review* (23:2), pp. 242-266.

Ouchi, W.G. 1978. "The Transmission of Control through Organizational Hierarchy," *Academy of Management Journal* (21:2), pp. 173-192.

Ouchi, W.G. 1979. "A Conceptual Framework for the Design of Organizational Control Mechanisms," *Management Science* (25:9), pp. 833-848.

Pflügler, C., Wiesche, M., and Krcmar, H. 2018. "Subgroups in Agile and Traditional IT Project Teams," *Hawaii International Conference on System Sciences*, Waikoloa Village, USA.

Przybilla, L., Wiesche, M., and Krcmar, H. 2018. "The Influence of Agile Practices on Performance in Software Engineering Teams: A Subgroup Perspective," in: *ACM SIGMIS-CPR* Buffalo, USA.

Riemer, K., and Klein, S. 2008. "Is the V-Form the Next Generation Organisation? An Analysis of Challenges, Pitfalls and Remedies of ICT-Enabled Virtual Organisations Based on Social Capital Theory," *Journal of Information Technology* (23:3), pp. 147-162.

Ross, J.W., Sebastian, I., Beath, C., Mocker, M., Moloney, K., and Fonstad, N. 2016. "Designing and Executing Digital Strategies," in: *International Conference on Information Systems*. Dublin, Ireland.

Runyan, R.C., Huddleston, P., and Swinney, J.L. 2007. "A Resource-Based View of the Small Firm: Using a Qualitative Approach to Uncover Small Firm Resources," *Qualitative Market Research: An International Journal* (10:4), pp. 390-402.

Sarker, S., Xiao, X., and Beaulieu, T. 2013. "Guest Editorial: Qualitative Studies in Information Systems: A Critical Review and Some Guiding Principles," *MIS Quarterly* (37:4), pp. iii-xviii.

Schermann, M., Wiesche, M., and Krcmar, H. 2012. "The Role of Information Systems in Supporting Exploitative and Exploratory Management Control Activities," *Journal of Management Accounting Research* (24:1), pp. 31-59.

Schlauderer, S., Overhage, S., and Fehrenbach, B. 2015. "Widely Used but Also Highly Valued? Acceptance Factors and Their Perceptions in Water-Scrum-Fall Projects," in: *International Conference on Information Systems*. Fort Worth, USA.

Sebastian, I.M., Ross, J.W., Beath, C., Mocker, M., Moloney, K.G., and Fonstad, N.O. 2017. "How Big Old Companies Navigate Digital Transformation," *MISQ Executive* (16:3), pp. 197-213.

Shaft, T.M., and Vessey, I. 2006. "The Role of Cognitive Fit in the Relationship between Software Comprehension and Modification," *MIS Quarterly*, pp. 29-55.

Towry, K.L. 2003. "Control in a Teamwork Environment—the Impact of Social Ties on the Effectiveness of Mutual Monitoring Contracts," *The Accounting Review* (78:4), pp. 1069-1095.

Tripp, J.F., Riemenschneider, C., and Thatcher, J.B. 2016. "Job Satisfaction in Agile Development Teams: Agile Development as Work Redesign," *Journal of the Association for Information Systems* (17:4), pp. 267 – 307.

Wagner, H.-T., Beimborn, D., and 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), pp. 241-272.

West, D., Grant, T., Gerush, M., and D'silva, D. 2010. "Agile Development: Mainstream Adoption Has Changed Agility," *Forrester Research* (2:1).

Wiedemann, A., and Weeger, A. 2017. "Developing Intellectual Capital within IT Teams: A Literature Review," in: *European Conference on Information Systems*. Guimarães, Portugal.

Wiener, M., Mähring, M., Remus, U., and Saunders, C.S. 2016. "Control Configuration and Control Enactment in Information Systems Projects: Review and Expanded Theoretical Framework," *MIS Quarterly* (40:3), pp. 741-774.

Wiesche, M., and Krcmar, H. 2014. "The Relationship of Personality Models and Development Tasks in Software Engineering," *ACM SIGMIS-CPR*, Singapore, Singapore: ACM, pp. 149-161.

Yin, R.K. 2009. *Case Study Research: Design and Methods*. Thousand Oaks, CA: Sage Publications Inc.

A Control-Alignment Model for Product Orientation in DevOps Teams– A Multinational Case Study

Completed Research Paper

Anna Wiedemann
Neu-Ulm University of Applied Sciences
Wileystraße 1, 89231 Neu-Ulm
anna.wiedemann@hs-neu-ulm.de

Manuel Wiesche
Technical University of Dortmund
Martin-Schmeißer-Weg 12,
44227 Dortmund
manuel.wiesche@tu-dortmund.de

Jason B. Thatcher
University of Alabama
348 Alston Hall, AL 35487
jbthatcher1@culverhouse.ua.edu

Heiko Gewald
Neu-Ulm University of Applied Sciences
Wileystraße 1, 89231 Neu-Ulm
heiko.gewald@hs-neu-ulm.de

Abstract

Changes in IT organization and technology environments make it necessary to adapt and review how mission critical IT functions align with firm strategy. IT functions increasingly use cross-functional teams to manage the lifecycle of digital solutions. As cross-functional teams begin to alter how we develop and maintain software, they may also result in control–alignment misfits that diminish the efficacy of functional project and operations controls. In this paper, we examine how the integration of product-oriented cross-functional teams challenges and transforms the IT function. We examine multinational cases of cross-functional DevOps teams in large companies. Rooted in control and alignment theory, we present a "bivariate management model" for aligning cross-functional teams and illustrate how shared domain knowledge becomes a critical enabler of product-oriented DevOps teams. This model contributes to literature by offering insight into how better design control elements of DevOps teams embedded in the IT function.

Keywords: IS control, shared domain knowledge, alignment, product orientation, DevOps, multinational case study

Introduction

Advances in information technology (IT) have paved the way for new forms of governance arrangements such as digital platforms (Tiwana et al. 2013), product networks, and for managing digital services (Ross et al. 2016). A key concern for IT functions is to combine flexibility and control in software development and delivery processes (Lamb and Kling 2003). Control in software development focuses on formal and informal project control mechanisms, whereas control in software operation concentrates on infrastructure and running software applications. Hence, software development and software operations control have different and opposing aims. In response to the specific challenge of aligning control mechanisms for development and operational activities within IT departments (Cram et al. 2016b), IT governance has moved from governing functional silo units toward managing cross-functional teams. Creating control capabilities in internal IT teams is necessary to achieve consonant communication and to coordinate autonomous self-managing teams responsible for planning, building, and operating digital solutions (Gregory et al. 2018).

More and more companies are establishing internal cross-functional IT teams to ensure the alignment of information systems (IS) and organizational objectives in IT decisions to fulfill new requirements (Gregory et al. 2018). However, maintaining strong communication and collaboration among employees with different backgrounds has proven elusive for managers (Kirsch 2004). Thus, communication and integrative collaboration are needed to achieve congruence an synergy effects within the IT department (Onita and Dhaliwal 2011). Many firms have accelerated this process by adopting software delivery concepts, such as DevOps (development and operations) to coordinate and align disparate communications and opposing goals of development and operations (Krancher et al. 2018).

By combining software developers' and software administrators' perspectives into one self-reliant team (Krancher et al. 2018), firms seek to encourage bidirectional governance and knowledge sharing (Tiwana et al. 2013). There is an inherent conflict of interest between software developers tasked with providing new software code and software operators who require stable software with minimal change (Wiedemann and Wiesche 2018). Aligned control mechanisms generate consensus within cross-functional teams and avoid misalignment at an early stage. Extant literature highlights that knowledge is a necessary element of organizational control (Ouchi 1979; Tiwana and Keil 2007). Notably, for internal IT units to effectively manage the software delivery lifecycle as self-reliant team, there must be an alignment between the IT knowledge and business knowledge of different departments (e.g., software development and software operations).

This paper aims to provide a better understanding of how firms encourage the alignment of their software development and operations activities. Furthermore, this study seeks to develop a model that explains the interplay between such activities within firms. Existing research focuses on control–alignment in different software development project setups e.g., Cram et al. (2016b) or on infrastructure governance and control, e.g., Yoo et al. (2010). Perhaps due to recent developments in relative product-oriented software, few studies have examined how changing control and governance structures affects product orientation in cross-functional IT teams. To fill this gap, we concentrate on the following research questions: *How does the implementation of cross-functional teams transform control in development and operations? How does shared domain knowledge influence alignment between software development and operations control elements?*

For answering our research questions, we conducted a multiple case study and used a partial portfolio Grounded Theory approach to analyze the data (Wiesche et al. 2017; Yin 2018). Conceived in the late 2000s, DevOps facilitates systematic collaboration between software developers and software operations to achieve faster time to market for deploying and operating new software features for digital solutions (Hemon et al. 2018; Wiedemann et al. 2019). DevOps enables the quick delivery of stable software through continuous delivery, continuous deployment, and solving disagreements between developers and operations through shared responsibility for software delivery processes (Kim et al. 2016). We examine cases in different countries and continents to gain broad insights into cross-functional DevOps teams, focusing of how concepts of control from IS research streams, such as software development and software operations, shed light on managing DevOps teams. Based on our field work, we present evidence of misfits between traditional notions of control and present a model of bivariate management for achieving alignment.

The remainder of this paper is organized as follows. We briefly review the IS control and alignment literature and note relevant research gaps. After describing our research method, we detail our multinational multiple case study and present the results of our analysis of the empirical data. We conclude the paper with a discussion of implications for research and practice.

Software Development and Software Operations Control in DevOps Teams

DevOps is often considered as an extension of agile software delivery, and also includes operations and maintenance activities (Hemon et al. 2018). DevOps bridges functional silo software development and operations units by integrating cross-functional teams. These teams are responsible for planning, building, and running system processes delivery lifecycles of digital solutions. The integration of cross-functional teams enables faster time to market of new software features through a continuous delivery technology and with higher quality as well as stability through the integration of the operations and software planning phases (Wiedemann et al. 2019). The control mechanisms employed by DevOps teams need to be transformed to enable high performance in a cross-functional working environment (Hemon et al. 2018), but research in this area is rare. Control is a common theme in the software development and IT alignment literature. Control refers to all actions taken to align the behavior of an individual or group with organizational goals (Ouchi 1979; Wiener et al. 2016). Cram et al. (2016a) apply this definition to IT, defining IS control as the influence of a person's or group's behavior to achieve objectives regarding the design, development, operation, usage, and management of IS. The software development literature leverages the concept of control to understand how to better design teams and organize projects (Ouchi 1979; Wiener et al. 2016).

In the following, we review the notions of control found in extant literature and applied to software development and operations. We analyze existing control literature of development and operations, being the two core elements of DevOps. Existing literature explains four control elements: measurement, evaluation, rewards and sanction as well as roles and relationships, that describe the dynamic of control changes during the software development phases (Eisenhardt 1985; Kirsch 2004). In addition, we analyze software operations, platform, and infrastructure literature, and mapped our findings from literature onto these control elements (detailed explanations below). These insights from existing research guided our research and theory development for answering the first research question: How does the implementation of cross-functional teams transform control in development and operations? Furthermore, we propose that shared domain knowledge ultimately aligns control between software operations and development in cross-functional teams. Hence, the second open question in this paper is: how does shared domain knowledge influence alignment between software development and operations control elements?

Development Control

At first, we investigated literature in the area of software development control. Because software development is one major part of DevOps, this literature stream guided our research about the transformation of control mechanisms in this area. Extant control literature differentiates between formal and informal modes of IS development projects (Jaworski 1988; Kirsch 1996). Formal control focuses on performance evaluation, where either input, behavior, or outcome is measured and rewarded. Informal control concentrates on people and social aspects through clan, and self-control modes (Eisenhardt 1985; Kirsch 1996; Mähring 2002). In controller-controlee relationships, these forms of control are often combined (Ouchi 1979) and used in portfolios of control mechanisms for supporting management practices in diverse contexts (Kirsch 1997). This research concentrates on control mechanisms that are used to exercise control. Existing studies of control in IS projects and software development concentrate on four control elements (see Table 1) as a basis for understanding changes in control. These control elements examine the mechanisms that improve the understanding of how control is exercised (Eisenhardt 1985; Kirsch 2004).

The first element is measurement that represents specifying and measuring control. Formal measurement presumes that behavior and outcome are explicitly defined and measurable, whereas informal measurement implies that norms, behavior, and values are implicitly defined and measured. Second, the control element of evaluation focuses on mechanisms for assessing performance and information exchange. Formal evaluation builds on specific information relating to behavior, and outcomes, and assesses whether the current situation leads to accelerating progress. Informal evaluation concentrates on norms, behaviors, and values that distinguish a functional relationship through socializing actions like discussions intended to lead to performance. Third, rewards and sanctions as an element based on measurement and evaluation include implicit and explicit gratification and improvement opportunities. The setting is formal when goals or desired behaviors are previously defined and communicated whereas an informal setting is determined whether a behavior is in line with group values and norms. Fourth, the element of roles and relationships is an extension of the three prior elements and refer to an individual's inclusion and relationships. In formal settings, dyadic relationships are in focus, whereas roles in an informal setting typically pertain to relationships among groups of people that depend on each other and committed to common goals (Kirsch 2004; Ouchi 1979; Persson et al. 2012).

Table 1. Elements of Project Control Adopted from Kirsch (2004, p. 378)

Element	Formal	Informal
Measurement	• Pre-specified and formally documented goals and/or behaviors are available • Control modes align the goals of controller and controlee • Goals and/or behaviors are measurable	• Few specified behaviors or procedures available • Implicit specification and measurement of group values and norms • Goals evolve over time • Desired end states result when individual behavior is consistent
Evaluation	• Information about rules, procedures, behaviors, and goals are exchanged • Information is exchanged in formal, written documents such as standard operating procedures or status reports • Evaluation assesses whether behavior is resulting in forward progress	• Information about norms, values and expectations exchanged • Socialization, training, discussions, dialogues and meetings serve as mechanisms of information exchange • Goal of evaluation is to build and foster collegial relationships characterized by common values and norms
Rewards and sanctions	• Based on following specified rules or achieving specified targets • Formal organizational mechanisms including pay, bonuses, promotion, or demotion	• Based on acting in a manner that is consistent with group norms and values • Mechanisms include group recognition and peer pressure
Roles and relationships	• Focus is usually on dyads • Controller and controlee are often in a formal superior–subordinate relationship or in a relationship that is consistent with the organizational hierarchy	• Often a work group or professional society • May be a clan, which is a group of individuals who are dependent on each other to accomplish their work and who are committed to achieving group goals

Operations Control

The second literature stream that guided this research is operations control because software operations are the second main part in DevOps teams. These extant investigations present insight of transforming operations control. Including operations tasks in cross-functional teams broadens the agility to software architecture and systems (Hemon et al. 2018). Hence, we propose that integrating operations and development control mechanisms will lead to the alignment of behavior in a cross-functional work environment. Operations control refers to how we shape changes and manage day-to-day activities of the information systems function. IT operations controlling is defined as the monitoring and controlling of IT services and infrastructure. For example, the IT infrastructure library (ITIL), a registered trademark, describes best practice for IT service management processes (Trusson et al. 2014). Operational IT failures of IT systems, e.g., software, networks, depict the presence of IT asset weaknesses considering control mechanisms over IT systems remains important (Benaroch and Chernobai 2017) because evolving digital infrastructure (Tilson et al. 2010) and platform technologies (Tiwana et al. 2013) shape the relevance of different control mechanisms. Integrating DevOps teams requires a wide spectrum of operations control mechanisms.

Digital infrastructure underpins contemporary open, heterogeneous, and evolving socio-technical systems and presents challenges for control in participating organizations (Koutsikouri et al. 2018; Tilson et al. 2010). Digital platforms offer services and standards to customers through automated coordination among stakeholders (Benaroch and Chernobai 2017) and influence management control responsibilities (Gregory et al. 2018). Specifically, research shows software-enabled control mechanisms (Yoo et al. 2010) and platform-based governance mechanisms (Gregory et al. 2018) support autonomous cross-functional teams. We refer to these perspectives as infrastructure control, which is defined as the influence on how users communicate with each other through infrastructure standardization. Infrastructure controls integrates rules embedded in technology as well as norms that shape how users communicate through IT assets within organizations.

System control refers to implementation, changes, and support of software applications. Such application-oriented changes are necessary to meet user demands during releases and after the deployment of new software. Systems control includes monitoring the accomplished tasks of the software (e.g., data flow, logging) for debugging activities and the resolution of emerging problems (e.g., incidents) (Edberg et al. 2012). To better understand how control is changing and enacted in cross-functional teams, we mapped the extant IS operations literature concerning infrastructure and systems control. The four control elements outlined above guided our literature review and we identified key mechanisms and the influence on operational control as summarized in Table 2.

Table 2. Elements of Operations Control Adopted from Literature

Elements	Infrastructure	System
Measurement	• Pre-defined interfaces for measuring data traffic • Goals are socio-technical (governing collaboration of different stakeholders.) • Measured by input and output parameters and integrated for measuring data traffic from heterogeneous components of an infrastructure (Koutsikouri et al. 2018)	• Pre-defined and formally documented objectives regarding every supporting application (service level agreements/ key performance indicators.) • Goals of software, incident tickets and problems (Nelson et al. 2000) • Speed/cost/value of software releases, ticket resolution
Evaluation	• Information of modular architecture (Yoo et al. 2010) • Information about large-scale infrastructure systems including multiple actors (Aanestad and Jensen 2011) • Evaluation of whether infrastructure ensures scalability (Constantinides and Barrett 2014)	• Information about systems/ application • Information exchange via support units/tickets (Nelson et al. 2000) • Evaluation of application stability and outage rate (Edberg et al. 2012)
Rewards and sanctions	• Based on new digital innovation that enable software-enabled infrastructure (Yoo et al. 2012)	• Alignment of actions with specific targets with mechanisms including pressure from other stakeholders, e.g., priorities (Edberg et al. 2012).
Roles and relationships	• Controller and controlee are often at the same organizational hierarchies with a versatile focus (Henfridsson and Bygstad 2013). • Network of human and non-human (e.g., technology) (Hanseth and Monteiro 1997).	• Often organized as work groups (support centers.) • Integrated or separate IT unit (Edberg et al. 2012).

Shared Domain Knowledge as Antecedent for Alignment

Extant literature on the alignment of software development and operations focuses on how the context—specifically, shared domain knowledge—shapes the goals of the IT function with the goals of the broader organization (Carter et al. 2011; Reich and Benbasat 2000; Tiwana 2012). Shared domain knowledge is seen as an antecedent of alignment (Reich and Benbasat 2000). To fully understand what drives changes in control of cross-functional teams, we need to consider shared domain knowledge. Knowledge is considered as shared when persons have knowledge in the domain of an area outside of their own specialized activities and domain (Carter et al. 2011; Tiwana and Keil 2007). Shared domain knowledge can be symmetrical (Nelson and Cooprider 1996) or asymmetrical, wherein one individual or unit has greater shared domain knowledge than the other. Some literature examines control and alignment in terms of "peripheral knowledge", which refers to knowledge outside of the specialized domain of activity (Tiwana 2012; Tiwana and Keil 2007). Peripheral knowledge contracts the concept of departmental specialization (Ouchi 1979) because it weakens departmental specialization to achieve other benefits (Tiwana 2012).

Existing research of shared domain knowledge distinguishes between the firm (IT unit's business domain knowledge and clients' technical knowledge) and project level (e.g., IT units knowledge and partner knowledge in outsourcing arrangement) (Nelson and Cooprider 1996; Tiwana 2012). However, research on shared domain knowledge within IT units is rare, which is surprising given that software development and software operations units typically have very different knowledge specialization (Edberg et al. 2012). This suggests a need to further explore shared domain knowledge in software development and delivery because successful cross-functional DevOps teams are likely to require team-level knowledge sharing.

In sum, our research builds on the presented existing literature in the areas of control and alignment. The aim is to understand how shared domain knowledge can accelerate control elements of cross-functional teams, align developers and operations staff, and foster product orientation. This is important because cross-functional teams typically do not operate in a conventional command and control structure (Persson et al. 2012). Because our goal is to derive recommendations of product-oriented control and alignment configurations, we leverage existing research on control balancing (Cram et al. 2016b) and the combination of different portfolios of control modes and elements in project environments (Kirsch 2004; Kirsch 1997), focusing on product orientation. Specifically, we see a need to merge control elements from different IS control research streams but also adopt new ones.

Research Method

To examine control dynamics in cross-functional teams, we apply a multiple case study and used a partial portfolio Grounded Theory approach to understand the setup of cross-functional teams and the control mechanisms necessary to achieve product orientation. This inductive research design allows us to answer "how" and "why" questions about the DevOps phenomenon based on analysis of real-life situations. Therefore, the overall objective of this study is to develop rather than test theory (Birks et al. 2013). As a first step, we analyze the cross-functional product-oriented team phenomenon by applying techniques from Ground Theory Methodology (Birks et al. 2013; Wiesche et al. 2017) and derive the new theoretical model of bivariate management that becomes central to in our theory development.

Research Site

Our multinational case design includes incumbent "brick and mortar" firms in Canada, Germany, USA, Africa, and Australia. The transnational nature of this study arose through respondents to the call for research on control in global contexts due to of its impact on control choices. The international context influences control through differences among time zones, locales, and culture (Persson et al. 2012). The transnational nature of this study was necessary because software development has moved from traditional IT silos toward integrating cross-functional teams like DevOps across the globe; thus gleaning insight into this movement requires case studies of best practices in firms in different national contexts. Furthermore, we want to identify whether mechanisms and learning possibilities used to spread knowledge and align DevOps teams differ across continents, e.g., Africa, America, Europe, and Australia. Therefore, we searched for examples around the globe that have already implemented DevOps teams to a certain degree.

The firms in our study range in size from 100–250 to > 100,000 employees and are in different industries (see Table 3). All firms use the DevOps method to a significant degree and have internal DevOps teams responsible for a minimum of one IT product. We identified several cases at practitioner conferences at which these companies held presentations and were positioned as outstanding examples or role models for DevOps. We sampled for firms that operated globally, so that we could investigate transboundary learning processes. While all research sites have integrated DevOps-oriented teams, each case constitutes a distinct industry, product line, and customer base. Table 3 presents a brief overview of each participating firm.

Table 3. Primary Data - Multinational Multiple Case Studies and Expert Interviews[1]

Primary data 35 interviews				
Cases and context		IT department	Interviewees	Team description
USA	**CongloCo** A large diversified industrial corporation that is operating worldwide in different sectors and industries, with more than 100,000 employees	IT department is mainly traditionally oriented, with some DevOps teams	Four (Manager, Senior IT Director, two DevOps Engineers)	Seven DevOps team members responsible for frontend service of a portal for customer request for new technology; the team uses Kanban principles
	ComCo A worldwide operating telecommunication provider with a broad range of services and 1,501-5,000 employees	IT function is transforming from traditional to DevOps orientation	Four (Senior Vice President, two Executives, Principal Architect)	Four DevOps team members responsible for cloud-based platform to support other teams; the team uses Kanban principles
Canada	**InsurOrg** One of the leading insurance that offers insurance services in many different areas such as car, home, and life, and provided by 50,001–100,000 employees	High level of traditional orientation and starting to integrate DevOps teams	Three (Assistant Vice President, Senior DevOps Manager, DevOps Manager)	12 DevOps team members responsible for integrating automation processes within the company and other teams; the team uses Scrum principles
	TechCo A technology firm that provides tools for software development and delivery for automation and integration (100–250 employees.)	IT function has some internal DevOps teams and has some traditional services	Four (DevOps Team Lead, three Team Members)	Six DevOps team members responsible for developing and managing integration hub for external customers; the team uses Kanban principles
Germany	**GrocCo** A leading food and retail chain company with more than 100,000 employees mainly in Europe; its sales channels are stores and an online shop	The company started transforming some projects to DevOps	Three (Division Manager, Manager and DevOps Team Member, Manager)	Six DevOps team members responsible for configuration management tool and supporting other teams; the team uses Scrum principles
	ServiceOrg An Internet company with more than 1,000 employees; helps end customers identify, compare, and buy products	The complete IT function is managed with the help of DevOps	Five (DevOps Team Lead, two DevOps Team Members, two Site Managers)	Four DevOps team members responsible for product providing web services to end-user; the team uses Kanban principles
	TravelCo A well-known online travel agency with more than 1,000 employees in Germany. Serves customers mainly online	One DevOps team for their platform and some DevOps services	Four (DevOps Team Lead, three DevOps Team Members)	Six DevOps team members responsible for online shop platform; the team uses Scrum principles
Australia	**ResearchOrg** Research consulting institution with more than 250 employees; conducts research in different areas e.g. health	IT function has different teams, e.g., innovation teams and some DevOps teams	Two (DevOps Team Lead, DevOps Engineer)	Nine DevOps team members responsible for the platform and managing network data structures; the team uses Scrum principles

Africa	**FinCo** A leading bank in Africa with more than 50,000 employees started agile transformation and is turning toward DevOps; offers various types of banking products, apps and online services	IT function started as agile and is now turning toward DevOps with the help of the scaled agile framework (SAFE)	Six (DevOps Coach, two Solution Engineers, two Enterprise Architects, Team Lead)	Eleven DevOps team members responsible for digital credit card services; the team uses Scrum principles
7 Interviews with Experts and Thought Leaders				
• Five prominent DevOps thought leaders who have significantly impacted the practice community through international lectures and publications (all from USA) • One system administrator with over 35 years' experience in a global technology company in USA • One manager with extensive DevOps research and practical expertise in Australia				
>300 pages of documentation	We transcribed all interviews and used memos during the coding process and data analysis, and analyzed company presentations and publications, which fostered discussions with the research team and theory development.			
[1] We used pseudonyms to protect firm identities.				

Data Collection

We identified leading experts and potential case study participants through targeted conversations and onsite presentations. Our cases include one cross-functional team per interview and several interviews per firm. We conducted semi-structured interviews with open-ended questions face-to-face, by video conferencing, and by telephone (Urquhart et al. 2010). We began each interview with introductions and asking questions about their DevOps-related experience and current relevant responsibility. The main focus of the interview consisted of questions about control and alignment between development and operations, featuring questions such as: What are the benefits of various DevOps team structures? What do you consider as essential components of a DevOps team? What are the similarities and differences between managing IT development and IT operations activities in the team? What are the benefits of various mechanisms to control the DevOps team?

The primary dataset consists of 35 interviews with DevOps team members, DevOps leaders, and senior managers from nine sites in Germany, USA, Canada, Africa, and Australia. After each interview, we documented the insights in analytic memos and the interviews were transcribed. On average, the interviews lasted 54 minutes. To achieve a better understanding of the research phenomenon, we interviewed five additional prominent DevOps thought leaders, a highly experienced system administrator, and a leading scholar and practitioner on the digitization of infrastructure. In sum, we conducted 42 interviews. Using multiple data points helps improve validity and mitigate potential bias (Yin 2018). To this end, parallel to our interviews, we collected complementary secondary data from, e.g., DevOps conferences, onsite visits to companies, listening to academic talks, and from the practical literature, following scientific guidelines. The Appendix presents information about the secondary data collection and analysis in addition to Table 3. Having multiple investigators in the analysis adds objectivity and enhances creativity to find new contributions (Eisenhardt 1989). One person conducted the interview and took notes. Afterward the interview's key insights were discussed and summarized within the research team and additional questions for future interviews were formulated (Urquhart et al. 2010).

Data Analysis

We employed a partial portfolio Ground Theory approach for data analysis (Wiesche et al. 2017). We focused on exploring mechanisms used by controllers and controlees of cross-functional DevOps teams and how shared domain knowledge enhances alignment. Furthermore, we examined product-oriented alignment and controlled transformation in organizations. Through constant comparison, we continuously compared the instances of both data and literature to identify categories, thereby contributing to theory development. We used iterative coding and constant comparison techniques to develop open and selective codes through several iterations (Birks et al. 2013; Urquhart et al. 2010; Wiesche et al. 2017). Subsequently, we discussed the findings of coding within the research team to identify the theoretical model (Wiesche et al. 2017). Additionally, we leveraged memo-taking in structuring our coding procedure. We started with open line-by-line coding of all our transcripts based on over 300 interview codes clustered into initial open codes regarding control and alignment. We then identified key patterns through selective coding and linked them to existing literature. These two core concepts **controlled-by-competency** and **controlled-by-invisibility** describe the control concepts of our new theoretical model, which we designate as **bivariate management**. Controlled-by-competency was derived based on the following concepts: leadership transformation, expansion of ownership, and human resource development. The second core concept, controlled-by-invisibility, based on the concepts of supervision, work relief, democratization, and standardization. Detailed descriptions of these categories, codes, and data excerpts are presented in the following section.

Findings

Our method allowed us to identify control mechanisms implemented by cross-functional teams, which we describe as a "bivariate management" approach. Bivariate management is a set of control mechanisms that enables the alignment of different mindsets, in our case development and operations, by controlled-by-competency and controlled-by-invisibility. Control-by-competency yielded insight into the importance of the human-to-human relationships for control of cross-functional DevOps teams, which requires and results in a high level of competency. In many DevOps teams, members learn and guide each other and evolve into mutual coaches. Control-by-invisibility characterizes control by human-to-technology relationships within cross-functional teams. For example, DevOps teams using software may implement a software-enabling architecture (Yoo et al. 2012) that facilitates controlling of team-related processes. Therefore, the teams use many invisible control mechanisms.

Bivariate Management Enables the Formation of Controlled-by-Competency

In DevOps arrangements, we found that leadership style differ from traditional software development teams. On DevOps teams, team leads or managers need **leadership transformation**, to serve as leaders that can motivate and enable people. *"In terms of the leadership style it is more focusing on the people and figuring out what they need and helping […], it is more about training and education"* (DevOps Manager, InsurOrg). Moreover, changes required by DevOps leads to a decentralization of responsibilities necessary to bridge development and operations work in one joint team. *"The level of involvement you have to do sometimes to see the way forward. It takes that lead to showing everybody else what we are going to do to make everything better*" (Chief of Staff, ComCo). In transformational leadership style, leaders serve as stewards and flat hierarchies are common forms of DevOps control mechanisms identified by our research.

In DevOps teams, the **broadening of ownership** by overtaking new tasks requires greater competency to master new tasks and capabilities. Organizations give responsibility to their DevOps teams, e.g., for choosing the right technology for their products. This makes the team accountable for the processes of the software delivery lifecycle, e.g., when a team member deploys a new piece of code into a production environment, the team colleagues will be in charge if something is not running smoothly. We recognize accountability as a new control mechanism within the team as well as from the team leadership perspective. One manager said, *"If you have to pay for yourself, you are much more cautious than if you know there is someone else who will fix it for you"* (ServiceOrg). Sometimes, only one person is responsible for managing products or supporting other teams: *"DevOps can either be dedicated people in teams from a specific department or it can be done by individual software engineers"* (DevOps Team Lead, ResearchOrg).

Effective DevOps teams **develop human resources**, by encouraging the learning of new competencies, skills, and knowledge. To better perform essential DevOps tasks, people have to share their expert knowledge in the area of development or operations within the team. *"The main thing has to do with understanding more about the operational space and what that skill set entails to better asses and coach individuals […]. I don't think that adding operations in development is drastically different; it's simply leveraging a lot of the best practices that we already knew from development from leadership and just expanding those into the operation space"* (Executive Director, ComCo). Bridging development and operations activities within the team requires members to be willing and able to learn new things.

Bivariate Management Leads to the Formation of Controlled-by-Invisibility

All the DevOps teams use automation tools and architecture to organize their work (**work relief**) and minimize the need for manual work. A higher degree of automation leads to fewer manual failures. Developers are responsible for the failures they made while writing a new piece of code. Hence, they want to prevent that failures appear and strengthen their mindsets for higher accuracy of source code. *"We do between four to five thousand changes per month. If they are automated, there's less risk of people making a mistake. The more we automate, the better the availability"* (Vice President, InsuOrg). Consequently, team members rely on automated tools and technology to avoid failures during the deployment of new software features.

While DevOps enables heterogeneous software development methods, our findings indicate the necessity for some **standardization**, as it helps manage huge organizations that have to pass external audits and maintain a high level of security. When necessary, DevOps teams create standards to encourage consistency. For example, one manager of ServiceOrg mentioned *"if anyone says I need ABC or any technology, then my standard question is: what can it that we do not already have [...]? I'm really happy if somebody says that, 'we would like to go to ABC because [this tool is no longer supported]', then I say, well, let us think about how we can do it together and then establish it as new standard."* Integrating decision making processes in the team fosters responsibility, but firms restrict the selection of new technology using guidelines to achieve formalization at a negotiable level.

Our key informants explained that using cloud and open infrastructures enables quick software releases of small software code with zero downtime required re-thinking controls. *"You can deploy with zero downtime with the right infrastructure"* (Manager, CongloCo). This requires scaling IT resources to meet varying demands on IT infrastructure. Our informants described how micro services, enabled by container technology or the use of virtual machines from public cloud service providers, offered broader access to developers seeking to commit changes to code. Hence, allowing the DevOps team members to use, e.g., open source technology calls for the **democratization** of IT access regulations in incumbent firms.

In offering greater access to make changes, DevOps methods resulted in a greater need to monitor and **supervise** changes made by team members. A team member mentioned that *"we check the applications logs to see if the request is handled correctly. So far, we have always been able to identify the problems"* (TravelCo). Key informants reported having to create and implement metrics for measuring and observing application performance, log files, new code, and failures. Many organizations created monitoring technology to continuously monitor processes and changes made by DevOps teams.

Shared Domain Knowledge Enables Bivariate Management

When considering bivariate management, we found that shared domain knowledge enables long-term alignment and provides the base for efficient control-by-competency and control-by-invisibility. More precisely, our findings indicate that the combination of a high level of control-by-competency, control-by-invisibility, and shared domain knowledge is required for achieving high performance in cross-functional teams. DevOps team members' knowledge moves from high specialization toward generalization. *"We usually have to pick up quite a lot of knowledge, and now this [knowledge] is cross-functional"* (Team Lead, ResearchOrg). For controlling DevOps teams, knowledge of both development and operations needs to be shared and adopted by all stakeholders. Our informants described how DevOps teams acquired knowledge from many different sources. For example, they look within their country to digital leaders, e.g., the cases in USA mentioned that they are interested in looking at GAFA (Google, Apple, Facebook, and Amazon) companies (Gregory et al. 2018). Additionally, our interviewees mentioned that they search for knowledge spanning other industries and countries: *"When you see what Toyota did, they did a combination of having folks be really good at those particular tasks but also learn faster. So, you had this kind of organizational design stamped out in a lot of places which focused on resource efficiency"* (Senior Vice President, ComCo). Managers also study other cases in the same industry. Case FinCo in Africa investigated the prominent case of ING Diba from The Netherlands and searched for learning opportunities.

The surveyed organizations encouraged learning and acquiring knowledge, investing a lot of effort to educate, train, and socialize DevOps team members to have a learning mindset and readiness to share their knowledge with their peers. One interviewee explained that s/he trains colleagues in the area of Kubernetes, an open-source technology for managing container-applications. *"Like even today we had a little Kubernetes session […]. A lot of people are interested in learning Kubernetes, and a couple of people on our team, myself and another engineers, know enough to help them and say 'here's what you need to get started"* (Team Member, TechCo). We found that controlled, aligned DevOps teams have high levels of shared knowledge and product orientation. Because of their deep understanding of the product, the DevOps teams actively engaged in collecting and sharing information with outside teams or open source communities engaged in producing similar products. *"Now people are you using a lot of open source packages, right? For example, even if I create a new program, I wouldn't write every piece of code, I would pull in different packages that other people have created"* (Team Member, TechCo).

Misfits in Executing Bivariate Management

Bivariate management is not always performed under ideal conditions resulting in discrepancies in controls and products when stakeholders build workarounds. We refer to these discrepancies as "misfits" in control–alignment product orientation. Misfits in control and alignment arise as a result of local differences in technology availability and also through discrepancies in cultural and local arrangement. For example, comparing FinCo from Africa with cases from USA and Germany, huge differences appear. The availability of technology and infrastructure in Africa is a challenge for FinCo: *"Global research is paramount, and applying best practices in a local context is another. In other words what works in the US and European markets doesn't always work in Africa. You have to take research with a pinch of salt and apply localized parameters and restrictions […] we have got very unequal society. Some people don't have running water and electricity so how are they going to have a high-speed fiber line? Our solutions have to cut across these kinds of different personas and we have to deliver IT solutions in a very different way because of the type of inequality"* (Enterprise Architect, FinCo). Some public cloud solutions that are standard in USA and Germany are simply not available in Africa. Hence, FinCo builds workarounds and use, e.g., open source technology and informs themselves on how to manage this problem.

Some identified misfits in control and alignment arose from conflicting approaches of software development in established firm processes. Some of the firms whose employees we interviewed have worked in third-tier support functions for years and made deep investments in ITIL as a means to achieve a high level of stability for running software products. Integrating DevOps into existing IT functions challenged traditional notions of IT service management such as incident and problem management. Companies used to work with ITIL have carefully structured and rigid approaches to IT support because ITIL *"defines […] processes like: strategy design, transition and operations, and those processes are unfortunately the kind of the antithesis of DevOps*." However, combining DevOps and ITIL can lead to a *"problem where folks have conflicting goals, right? They have goals of speed versus goals of stability. We really want to align those goals […and] we had to overcome any problems that ITIL present us"* (Senior Vice President, ComCo). At the time of our case studies, incumbent firms were still working to solve alignment problems by judiciously dissolving some rigid ITIL processes and integrating them into the DevOps teams to enable control.

Misfits in control and alignment also arise from large monolithic legacy systems. These systems were often developed to centralize computing and data management. Informants reported that modern DevOps infrastructure teams faced problems, because it is not easy, and sometimes not possible, to transform monolithic systems into micro services. *"API connects with a lot of legacy systems and normalizes the data to be very easy to work with for the portal. We use the gateway in the middle to kind of clean up all the legacy systems and provide an API to the portal. Sometimes the portal suffers [or] some legacy systems suffer maintenance issues or they've just been running for so long and it takes a lot of time to make a small change"* (Team Member, CongloCo).

Misfits in control and alignment also arose from team members' fears about applying a DevOps mindset. Leaders reported fear about changes, such as sharing intellectual property, leading to losing value and power within the firm. Moreover, team members are afraid of sharing knowledge or making mistakes because through open–source technology their specialized knowledge is shared within and outside of the company, thereby limiting their future opportunities. When leaders and managers felt challenged by DevOps, they deviated from bivariate management and reverted to traditional hierarchical structures: *"The biggest problem facing traditional enterprises right now is having the right leaders who really understand technology and the problems and second have the courage to actually make the changes that* *are required. One, it's hard to find the leaders and second, it's very hard to make these changes because someone has to give up something"* (Senior Vice President, ComCo).

Control Mechanisms of Bivariate Management

Our bivariate management model has several consequences. First, leaders must effectively align controls from software development and software operations to achieve design controls suitable for DevOps. Second, such control elements are most effective when team members share domain knowledge, which enhances the collaboration of team members with different specialist backgrounds. The results of our study indicate that in both major categories of bivariate management (controlled-by-competency and controlled-by-invisibility), shared knowledge is required to achieve a high level of cross-functional team performance resulting in product orientation. Third, bivariate management solves possible conflict situations at an early stage or before they appear. Merging the different goals and perspectives of team members with different backgrounds leads to achieving a common goal. Figure 1 depicts shared domain knowledge that serves as a catalyst for our identified control mechanisms resulting in a control–alignment tetrahedron. The tetrahedron depicted in the figure constitutes the optimal condition of aligned and controlled bivariate management.

Misalignment appears when control is not exercised simultaneously on a high level. For example, performing control with a great focus on controlled-by-invisibility leads to neglecting the controlled-by-competency part and vice versa. Shared domain knowledge enables the simultaneous execution of control, because it fosters the understanding of the different work that is conducted by team members and managers alike. The tips of the triangle show concurrently high levels of each variable, but can move when misalignment appears.

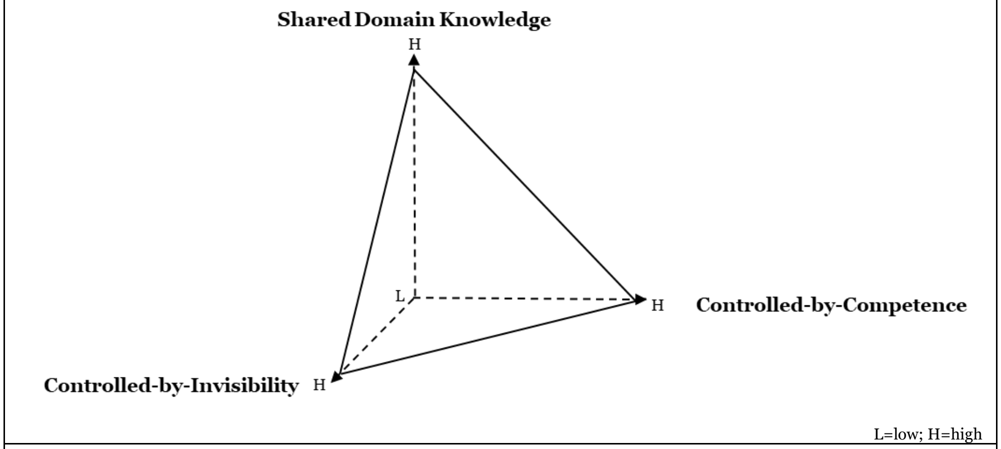


Figure 1. Bivariate Management as Control-Alignment Tetrahedron

As the next stage of our study, we present some examples for control elements in product orientation and extend and refine the theoretical framework of control elements presented by Kirsch (2004). Existing control frameworks tend to focus on either software development or operations control. For controlling cross-functional teams, our data analysis suggested a need to extend or go beyond existing understanding of controls. Our findings suggest that bivariate management needs to combine formal and informal with infrastructure and systems control to accurately describe how controls are enacted for DevOps teams. Table 4 below summarizes the core control elements and lists some examples for DevOps control mechanisms. Instead, of "Roles and Relationships", we call the fourth element "Relationships and Responsibility" to reflect the broader tasks assigned to DevOps team members.

Table 4. Control Mechanisms in Cross-Functional Teams

Elements	Formal and Informal ∞ Infrastructure and System
Measurement	Track benefits and improvement regarding commitments (e.g., learning goals) Enhance agile methods to operations (e.g., operations work on Kanban board) Allow time for unplanned work (e.g., software bugs) Implement metrics for measurement (e.g., user peaks) Set alerts with the help of monitoring (e.g., alters for on-call duty) Automate manual, recurred activities (e.g., autoscaling)
Evaluation	Information about company standards (e.g., obligatory audits) Evaluate value stream management of teams (e.g., essential process steps) Information about allowed tools and technology (e.g., public cloud) Information about API, self-service ability (e.g., API to get a new database) Communicate and make agreements within the team (e.g., agree team learnings)
Rewards and sanctions	Implement team milestones and goals (e.g., managing huge tasks) Build scalable, cost-friendly architecture with open-source (e.g., micro services) Resolve failures within the team (e.g., discuss problems in the team) Information about team members tasks and mutual learning (e.g., pair programming) Foster self-reliance (e.g., rewarding through self-motivation)
Relationships and responsibility	Integrate responsibility within team (e.g., managing software product) Make group decisions (e.g., new technology) Responsibility for software delivery lifecycle (e.g., "you build it, you run it")

Discussion

The findings from our study shed light on control and alignment in cross-functional DevOps teams and point to a novel bivariate management model. Such a bivariate model is necessary because the practices and expectations of DevOps teams require combining elements of software development and software operations control. Our key informants suggest that effective control of cross-functional teams requires blurring boundaries between principals and agents. This departs from classic controller–controlee relationships described in extant IT controlling literature (Tiwana et al. 2013). One reason for this departure is that cross-functional teams are product-oriented, more focused on outcomes and less focused on process. We found that DevOps teams are not only controlled by a team lead (controller–controlee), but that the team members also control themselves within a team and with the help of technology. To understand how DevOps teams achieve a high level of product orientation, we extend traditional software project control mechanisms and present a new model for managing product teams.

Our bivariate model distinguishes between controlled-by-competency and controlled-by-invisibility. We define controlled-by-competency as human-to-human control mechanisms that are related to human actions with regard to changes. Hence, we expanded on the IS software development project literature and found that formal as well as informal control mechanisms are necessary in DevOps teams. To achieve a high level of coordination among and integration of different autonomous parties requires control changes (Gregory et al. 2018; Tiwana et al. 2013). We identified three major categories of such change: leadership transformation, expansion of ownership, and human resource development. In contrast, we defined controlled-by-invisibility as human-to-technology relationships, including technological aspects used by team members to facilitate their work. These control categories are grounded in software operations literature and we have provided evidence that infrastructure as well as systems control are required for managing DevOps teams. Extant IT control literature focuses largely on project management (Wiener et al. 2016) and overlooks the important role of IT architecture. In distributed organizational arrangements, such as DevOps settings, control embedded in architecture becomes a more subtle and less costly means to control team members' work. Hence, our findings regarding controlled-by-invisibility are related to previous literature, underscoring architecture as a device effective for coordination and control (Tiwana et al. 2013). Eventually, we found that control mechanisms from development as well as operations are necessary as well as a high level of shared domain knowledge to achieve outperformance of DevOps.

Our model suggests that misfits result in tensions between DevOps team members or in firm processes. At the team member level, lack of shared knowledge creates tensions or inefficiencies. In traditional IT organizations, tensions appear because collaboration and communication between different IT units are rare. For example, software development teams typically have less control and knowledge about infrastructure components (Krancher et al. 2018). Similarly, software operations teams usually lack control and knowledge of coding and software development. This tension results in misunderstandings between employees that limit a firm's ability to implement new software. At the process level, software operations and development processes have different goals (e.g., ITIL = stability; agile = change) requiring different control mechanisms. Figure 1 indicates the optimal approach of control–alignment in product-oriented cross-functional teams. Neglecting one control part of bivariate management will lead to misalignment.

As with all research, this study has some limitations. First, we interviewed mainly DevOps experts from IT. Because cross-functional teams also frequently include business experts, future research should more thoroughly investigate the business perspective. Second, we concentrated on internal DevOps teams, indicating two major categories of controls. However, our interviews also suggest a need to consider other contexts, such as DevOps and outsourcing arrangements, possibly leading to a multivariate theory. Finally, given the qualitative focus of this research, quantitative validation of our insights and model would be valuable, such as a broad questionnaire of diverse DevOps teams across industries on control. Such an analysis is needed to assess the generalizability of our findings.

Implications for Theory

Our in-depth multinational case study of DevOps teams offers several theoretical contributions. Our findings present a novel bivariate management model that underscores the importance of two mechanisms (controlled-by-competency and controlled-by-invisibility) and shared domain knowledge as prerequisite for controlling DevOps teams. First, existing literature depicts self-organizing as a core element of DevOps, which is where no clear role separation exists between developers and system operators (Krancher et al. 2018). However, this view does not explain how to move from a project orientation toward product orientation, which is necessary to successfully deploy DevOps approaches. Our findings underscore the importance of human-to-human control relationships in making this transition because team members learn, and share knowledge with each other, and leaders transform into coaches. Hence, self-organized control hinges on developing product competency and sharing knowledge within the team. The model further develops the agile software development approaches toward agile operations. A high level of alignment between the different control mechanisms requires shared domain knowledge. Specifically, our results show that successful cross-functional teams must achieve a high level of shared domain knowledge to achieve alignment of development and operations control.

Second, existing research of software development control depicts insight into informal control like self- and clan control (Chua et al. 2012; Ouchi 1979), where different people work toward a common goal or self-monitoring of individuals behavior (Wiener et al. 2016). Research defines a clan as a group with a high level of social capital based on shared values (Chua et al. 2012). This is a very consensus-oriented form enacting control. Within product-oriented teams like DevOps teams, control is based on achieving results and resolving conflict . Developers and operations people have different characterizations and backgrounds, but our bivariate management model lays the groundwork for reducing conflict potential at an early stage by fostering collaborative communications and a results-driven working environment.

Third, our study contributes to infrastructure and platforms literature (Koutsikouri et al. 2018), by providing new insight into the overlooked impact of architecture on control. This paper presents a new form of control for DevOps teams that is embedded in human-to-technology relationships. Cloud platforms and container-oriented approaches enable the agile operations of systems. Our findings present novel insight into how architectural control constrains or enables DevOps team members. We believe that this insight deserves further investigation. For example, as architectures grow more scalable and interoperable, how will different hardware and software configurations shape the way DevOps teams work? How will global technological and architectural changes affect DevOps controlling? How will the opening or closing of digital borders (e.g., the cloud) or regulatory changes such as the General Data Protection Regulation change DevOps members' ability to collect, share, and integrate information? Future research should investigate suitable configurations of control elements in response to unexpected challenges.

Fourth, control elements are mainly discussed in project setups (Kirsch 2004; Persson et al. 2012). We broaden this framework to include operations control as described in the literature discussion. Moreover, this research presents how control elements regarding "formal and informal" and "infrastructure and system" interact with each other. Managing DevOps arrangements presupposes dynamics in control modes (Tiwana et al. 2013) to explain the movement to product orientation.

Implications to Practice

This work suggests that bivariate management is a useful means for controlling DevOps teams. It also underscores the need for managers to develop new skills, such as coaching and knowledge sharing, to empower DevOps teams. Further, it directs managers to create a culture of support among DevOps team members learning to guide each other, self-manage quality control, and co-define team goals. These skills will help managers improve product management skills in DevOps teams. Furthermore, we illustrate that managers must pay attention to the influence of infrastructure and architectural artifacts, which can both support and limit control and alignment of DevOps teams. Finally, we illustrate that DevOps require breaking down silos typical of software project work. We demonstrate that successful DevOps integrate team members from various backgrounds, create alignment through growing shared domain knowledge, and source/share knowledge with external groups such as open source communities.

Conclusion

Our findings introduce a bivariate management model to achieve product orientation, alignment, and control of cross-functional DevOps teams. Our multinational multiple case study makes significant contributions in several areas. We extend existing literature by explaining the controls necessary to effectively integrate software development and operations. Furthermore, we provide a better understanding of the potential of product orientation. In addition, our bivariate management model sheds light on resolving the control–alignment problem and move toward product orientation through shared domain knowledge. The primary outcome of our investigation is a richer understanding of a control–alignment model that helps explain fits and misfits of cross-functional DevOps teams. Our bivariate management model could be used by managers to ensure alignment of product-oriented DevOps teams. We illustrate how existing understanding of controls of software development and software operations can be leveraged to develop new forms of control to manage DevOps teams.

Acknowledgements

We thank the German Federal Ministry of Education and Research (BMBF) for funding this research under grant code 03FH005PX4.

References

Aanestad, M., and Jensen, T. B. 2011. "Building Nation-Wide Information Infrastructures in Healthcare through Modular Implementation Strategies," The Journal of Strategic Information Systems (20:2), pp. 161-176.

Benaroch, M., and Chernobai, A. 2017. "Operational IT Failures, IT Falue-Destruction, and Board-Level IT Governance Changes," MIS Quarterly (41:3).

Birks, D. F., Fernandez, W., Levina, N., and Nasirin, S. 2013. "Grounded Theory Method in Information Systems Research: Its Nature, Diversity and Opportunities," European Journal of Information Systems (22:1), pp. 1-8.

Carter, M., Grover, V., and Thatcher, J. B. 2011. "The Emerging CIO Role of Business Technology Strategist," MIS Quarterly Executive (10:1), pp. 19-29.

Chua, C. E. H., Lim, W.-K., Soh, C., and Sia, S. K. 2012. "Enacting Clan Control in Complex IT Projects: A Social Capital Perspective," MIS Quarterly (36:2), pp. 577-600.

Constantinides, P., and Barrett, M. 2014. "Information Infrastructure Development and Governance as Collective Action," Information Systems Research (26:1), pp. 40-56.

Cram, W. A., Brohman, K., and Gallupe, R. B. 2016a. "Information Systems Control: A Review and Framework for Emerging Information Systems Processes," Journal of the Association for Information Systems (17:4), pp. 216-266.

Cram, W. A., Brohman, M. K., Chan, Y. E., and Gallupe, R. B. 2016b. "Information Systems Control Alignment: Complementary and Conflicting Systems Development Controls," Information and Management (53:2), pp. 183-196.

Edberg, D., Ivanova, P., and Kuechler, W. 2012. "Methodology Mashups: An Exploration of Processes Used to Maintain Software," Journal of Management Information Systems (28:4), pp. 271-304.

Eisenhardt, K. M. 1985. "Control: Organizational and Economic Approaches," Management Science (31:2), pp. 134-149.

Eisenhardt, K. M. 1989. "Building Theories from Case Study Research," Academy of Management Review (14:4), pp. 532-550.

Gregory, R. W., Kaganer, E., Henfridsson, O., and Ruch, T. J. 2018. "IT Consumerization and the Transformation of IT Governance," MIS Quarterly (42:4), pp. 1225-1253.

Hanseth, O., and Monteiro, E. 1997. "Inscribing Behaviour in Information Infrastructure Standards," Accounting, Management Information Technologies (7:4), pp. 183-211.

Hemon, A., Monnier-Senicourt, L., and Rowe, F. 2018. "Job Satisfaction Factors and Risks Perception: An Embedded Case Study of Devops and Agile Teams," in: International Conference on Information Systems. San Francisco.

Henfridsson, O., and Bygstad, B. 2013. "The Generative Mechanisms of Digital Infrastructure Evolution," MIS Quarterly (37:3), pp. 907-931.

Jaworski, B. J. 1988. "Toward a Theory of Marketing Control: Environmental Context, Control Types, and Consequences," The Journal of Marketing (52:3), pp. 23-39.

Kim, G., Humble, J., Debois, P., and Willis, J. 2016. The DevOps Handbook. Portland, USA: IT Revolution Press, LLC.

Kirsch, L. J. 1996. "The Management of Complex Tasks in Organizations: Controlling the Systems Development Process," Organization Science (7:1), pp. 1-21.

Kirsch, L. J. 2004. "Deploying Common Systems Globally: The Dynamics of Control," Information Systems Research (15:4), pp. 374-395.

Kirsch, L. S. 1997. "Portfolios of Control Modes and IS Project Management," Information Systems Research (8:3), pp. 215-239.

Koutsikouri, D., Lindgren, R., Henfridsson, O., and Rudmark, D. 2018. "Extending Digital Infrastructures: A Typology of Growth Tactics," Journal of the Association for Information Systems (19:10), pp. 1001-1019.

Krancher, O., Luther, P., and Jost, M. 2018. "Key Affordances of Platform-as-a-Service: Self-Organization and Continuous Feedback," Journal of Management Information Systems (35:3), pp. 776-812.

Lamb, R., and Kling, R. 2003. "Reconceptualizing Users as Social Actors in Information Systems Research," MIS Quarterly (27:2), pp. 197-236.

Mähring, M. 2002. "IT Project Governance: A Process-Oriented Study of Organizational Control and Executive Involvement," Business Administration (15).

Nelson, K. M., and Cooprider, J. G. 1996. "The Contribution of Shared Knowledge to IS Group Performance," MIS Quarterly (20:4), pp. 409-429.

Nelson, K. M., Nadkarni, S., Narayanan, V. K., and Ghods, M. 2000. "Understanding Software Operations Support Expertise: A Revealed Causal Mapping Approach," MIS Quarterly (24:3), pp. 475-507.

Onita, C., and Dhaliwal, J. 2011. "Alignment within the Corporate IT Unit: An Analysis of Software Testing and Development," European Journal of Information Systems (20:1), pp. 48-68.

Ouchi, W. G. 1979. "A Conceptual Framework for the Design of Organizational Control Mechanisms," Management Science (25:9), pp. 833-848.

Persson, J. S., Mathiassen, L., and Aaen, I. 2012. "Agile Distributed Software Development: Enacting Control through Media and Context," Information Systems Journal (22:6), pp. 411-433.

Reich, B. H., and Benbasat, I. 2000. "Factors that Influence the Social Dimension of Alignment between Business and Information Technology Objectives," MIS Quarterly (24:1), pp. 81-113.

Ross, J. W., Sebastian, I., Beath, C., Mocker, M., Moloney, K., and Fonstad, N. 2016. "Designing and Executing Digital Strategies," in: International Conference on Information Systems. Dublin, Ireland.

Tilson, D., Lyytinen, K., and Sørensen, C. 2010. "Research Commentary—Digital Infrastructures: The Missing IS Research Agenda," Information Systems Research (21:4), pp. 748-759.

Tiwana, A. 2012. "Novelty-Knowledge Alignment: A Theory of Design Convergence in Systems Development," Journal of Management Information Systems (29:1), pp. 15-52.

Tiwana, A., and Keil, M. 2007. "Does Peripheral Knowledge Complement Control? An Empirical Test in Technology Outsourcing Alliances," Strategic Management Journal (28:6), pp. 623-634.

Tiwana, A., Konsynski, B., and Venkatraman, N. J. J. o. M. I. S. 2013. "Information Technology and Organizational Governance: The It Governance Cube," Journal of Management Information Systems (30:3), pp. 7-12.

Trusson, C. R., Doherty, N. F., and Hislop, D. 2014. "Knowledge Sharing Using It Service Management Tools: Conflicting Discourses and Incompatible Practices," Information Systems Journal (24:4), pp. 347-371.

Urquhart, C., Lehmann, H., and Myers, M. D. 2010. "Putting the 'Theory' Back into Grounded Theory: Guidelines for Grounded Theory Studies in Information Systems," Information Systems Journal (20:4), pp. 357-381.

Wiedemann, A., Forsgren, N., Wiesche, M., Gewald, H., and Krcmar, H. 2019. "Research for Practice: The DevOps Phenomenon," Communications of the ACM (62 8), pp. 44-49.

Wiedemann, A., and Wiesche, M. 2018. "Are You Ready for DevOps? Required Skill Set for DevOps Teams," in: European Conference on Information Systems. Portsmouth, UK.

Wiener, M., Mähring, M., Remus, U., and Saunders, C. S. 2016. "Control Configuration and Control Enactment in Information Systems Projects: Review and Expanded Theoretical Framework," MIS Quarterly (40:3), pp. 741-774.

Wiesche, M., Jurisch, M. C., Yetton, P. W., and Krcmar, H. 2017. "Grounded Theory Methodology in Information Systems Research," MIS Quarterly (41:3), pp. 685-701.

Yin, R. K. 2018. Case Study Research and Applications: Design and Methods. Los Angeles: SAGE Publication Inc.

Yoo, Y., Boland Jr, R. J., Lyytinen, K., and Majchrzak, A. 2012. "Organizing for Innovation in the Digitized World," Organization Science (23:5), pp. 1398-1408.

Yoo, Y., Henfridsson, O., and Lyytinen, K. 2010. "Research Commentary—The New Organizing Logic of Digital Innovation: An Agenda for Information Systems Research," Information Systems Research (21:4), pp. 724-735.

Appendix

Table A1. Secondary Analysis

Exploration and Sampling	
Initial Exploration	
Selection of suitable case study organizations using theoretical sampling (Eisenhardt 1989)	Exploration with 10 companies to prepare and develop the present multinational study in order to explore e. g., transboundary learning processes
Scientific Conferences and Presentations	
Active participation and presentation at scientific conferences in America, Asia, and Europe (e.g., AIS conferences) and research presentation in universities in Germany, USA, and Australia	• Exchange of experiences and knowledge with researchers in the field • Gathering insight about entering the field • Discussion of research ideas
Practical Insights	
Expert Interviews with Thought Leaders and Industry Experts	
Interviews with topic specialists and attending keynotes speeches held by them	Interviews with seven global DevOps thought leaders to gain insights into the novelty of the concept and better understanding practical implementation.
Attending Practice Conference	
Active participation and presentation of research at practice-oriented conferences and seminars in USA and Europe (e.g., DevOps Enterprise Summit)	• Exchange of experiences with practitioners and DevOps evangelists as well as knowledge sharing and presentation of prior research results • Identification of participants for second case study

EUROPEAN JOURNAL OF INFORMATION SYSTEMS
https://doi.org/10.1080/0960085X.2020.1782277

THE OPERATIONAL RESEARCH SOCIETY
Taylor & Francis
Taylor & Francis Group

EMPIRICAL RESEARCH

OPEN ACCESS Check for updates

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, Sabine Matook, Knut Rolland

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.

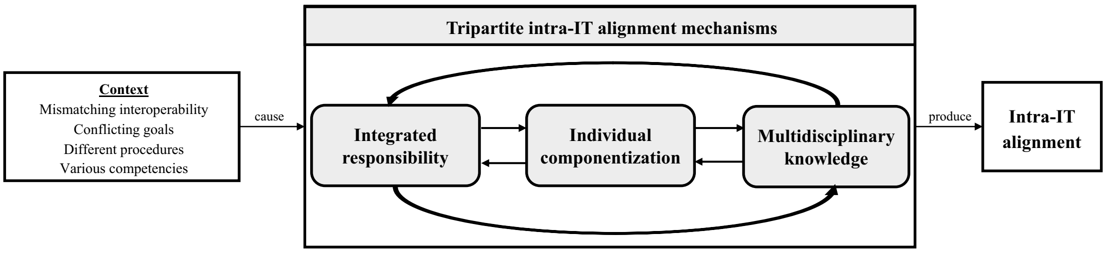

Context
Mismatching interoperability
Conflicting goals
Different procedures
Various competencies
cause
Tripartite intra-IT alignment mechanisms
Integrated responsibility
Individual componentization
Multidisciplinary knowledge
produce
Intra-IT alignment

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?