The DevOps Continuum: Walking the Shadowy Bridge from Information Systems Development to Operations

Michael Hüttermann
University of Cologne, huettermann@wiso.uni-koeln.de

Follow this and additional works at: https://aisel.aisnet.org/ecis2021_rp

Recommended Citation
https://aisel.aisnet.org/ecis2021_rp/78

This material is brought to you by the ECIS 2021 Proceedings at AIS Electronic Library (AISeL). It has been accepted for inclusion in ECIS 2021 Research Papers by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.
THE DEVOPS CONTINUUM: WALKING THE SHADOWY BRIDGE FROM INFORMATION SYSTEMS DEVELOPMENT TO OPERATIONS

Research Paper

Michael Hüttermann, University of Cologne, Cologne, Germany, huettermann@wiso.uni-koeln.de

Abstract

In recent years, enterprises have observed that a holistic approach to agile information systems development and a close integration of information systems operations is essential to maximize the probability of success, leading to the emerging DevOps phenomenon. While first studies have delivered first preliminary insights about DevOps, a foundational understanding of DevOps implementations and theoretical grounding of DevOps is still missing. To close this gap, we conducted a multiple-case study to explore a variety of characteristics of DevOps, and how these different characteristics influence its implementation in different contexts. We find that variations of DevOps implementations are gradual transitions on what we call a DevOps continuum. Based on this insight, we propose a conceptual model which fosters our understanding of the DevOps phenomenon, relates DevOps to existing theories, and identifies new paths for future research.

Keywords: DevOps, Continuum, Information Systems Development, Information Systems Operations

1 Introduction

In industry, the simultaneous consideration of both the development and the operations of information systems (IS) is central to the emerging phenomenon of DevOps (Hüttermann, 2012; Wiedemann, Forsgren, et al., 2019). DevOps is a portmanteau word of “development” and “operations”. Proponents of the concept argue that the two major information technology (IT)-related enterprise functions “IT development” (those that build IS) and “IT operations” (those that run and maintain IS) increasingly apply shared goals and use shared practices across both functions, bringing together team members from both development and operations, in order to implement information system development (ISD) in a comprehensive way (Lwakatare, Kuvaja, et al., 2016; Qumer Gill et al., 2018; Wiedemann, 2017). In essence, DevOps appears as a logical extension of agile information systems development (AISD) (Hemon-Hildgen, Rowe, et al., 2020), aiming to bridge development to its resulting IS product and its systems operations. Thus, companies applying DevOps streamline their IT development and IT operations to overcome well-known barriers and friction points between those two often siloed IT functions. DevOps aims to address major management concerns, including business productivity and cost reduction, IT and business alignment, and business agility and speed to market (Luftman et al., 2013).

Until now, scant studies have investigated DevOps because it is a very recent phenomenon. For example, existing research has proposed a tripartite model of intra-IT alignment, made up of individual componentization, integrated responsibility, and multidisciplinary knowledge, to shed light on how DevOps aligns development with operations (Wiedemann et al., 2020). Others have examined the orchestration of automation, sharing, and risk management in DevOps teams and their relationship to work conditions and job satisfaction (Hemon-Hildgen, Rowe, et al., 2020). While these first studies offer very valuable preliminary insights about DevOps implementations in specific settings, they focus
on particular aspects such as how alignment in DevOps teams can be achieved (Wiedemann et al., 2020) or how DevOps affects job satisfaction (Hemon-Hildgen, Rowe, et al., 2020). However, similar to AISD, the development of DevOps is primarily driven by industry. No formal definition for DevOps exists, and many practitioners argue that this is intentional because it allows teams and organizations to adopt a definition that works for them (Wiedemann et al., 2019). This lack of understanding, conceptualization, and theorizing is challenging for academic research and studies on DevOps.

In sum, we are missing a theoretical grounding of DevOps, and we are also lacking a foundational understanding of what leads to a specific DevOps implementation (i.e., how DevOps is executed), depending on a given context in industry, or what the effects of different DevOps implementations, configurations, or approaches are. Consequently, this leads us to the following research question for our study: “Which characteristics influence how DevOps is implemented in organizations?”

To answer the research question, we conducted an exploratory, multiple-case study in order to derive and suggest a model explaining DevOps. For IS research, we contribute to the nascent understanding of DevOps by providing an empirically grounded theoretical model. Since DevOps has a strong momentum in industrial practice, our results may offer valuable guidance to practitioners who execute (diverse) DevOps initiatives in their respective contexts. Next, we discuss the theoretical background and related work of our study. Afterwards, we describe our research method, present our main findings and discuss the resulting DevOps model. We conclude with highlighting avenues for further research.

2 Related Work and Theoretical Background

2.1 Origins of DevOps

In today’s industrial practice, applying AISD methods such as Scrum (Beck, 1999) or Extreme Programming (Schwaber and Beedle, 2002) is increasingly one promising way for enterprises to maximize the probability of ISD success (Dybå and Dingsøyr, 2009). Research revealed, however, that focusing solely on the development factors does by-and-large neglect the importance of operational factors, the resulting IS product, and its long-term operation and use (DeLone and McLean, 1992; Fitzgerald and Stol, 2017; Petter et al., 2013). This suggests that IS success (DeLone and McLean, 1992) can only be optimized if the organizational subunits of IT development and IT operations are integrated or aligned with each other (Wiedemann et al., 2020). Spanning development and operational factors, efficiency (“delivering the product right”) and effectiveness (“delivering the right product”) (Chandler Jr., 2018) contribute to IS success. Efficiency relates to how much IS functionality is shipped in a given time, whereas effectiveness focuses on the amount of customer value created by the shipped functionality (Bosch, 2019). Building on this understanding, industry often sees DevOps as an extension of AISD (Wiedemann, Forsgren, et al., 2019), which aims to optimize overall IS success and strives for better results of efficiency and effectiveness (Hüttermann, 2012).

Similar to its rising application in industry, DevOps has gained increasing attention from IS research (Sharp and Babb, 2018). With its roots in AISD, DevOps stems from the idea to extend the use of agile practices from ISD to IS operations, and thus bridges the often siloed IT functions of IT development and IT operations (Hüttermann, 2012). Due to the lack of specific research, models, and definitions, DevOps can describe different things, including team structure, success criteria (i.e., goals), concepts, or tooling (Qumer Gill et al., 2018).

2.2 First Steps Towards Conceptualizing DevOps

As a first step towards a conceptualization of DevOps, researchers have suggested dimensions such as culture, automation, lean, measurement, and sharing (CALMS) (Fitzgerald and Stol, 2017; Humble and Molesky, 2011; Wiedemann, Forsgren, et al., 2019). According to this understanding, DevOps aims to address cultural barriers between the development and operations functions, it strives for
automation of development and delivery processes across the functions, it emphasizes measurement and joint use of metrics across the functions, it applies lean principles such as removing waste, and it highlights the practice of sharing culture, goals, measures, automation and tooling between the functions. Extant research also argues that culture and measurement can be partially subsumed in the concept of sharing (Hemon-Hildgen, Rowe, et al., 2020), that sharing information and practices across functions leads towards finding a common ground and understanding of measures, and once sharing is practiced, automation of processes can be implemented more efficiently towards continuous delivery of software.

In industrial practice, automation and the related concept of continuous delivery, the production of software in short temporal cycles, resulting in building, testing, and releasing software with greater speed and frequency, often have the highest priority in order to satisfy customers through continuously providing valuable software (Fowler and Highsmith, 2001). A prerequisite for continuous software delivery is lean thinking to eliminate bottlenecks along the value chain across all contributing parties (Khan and Sarker, 2002; Poppendieck and Poppendieck, 2003; Wetherbe and Frolick, 2000), above all IT development and IT operations (Fitzgerald and Stol, 2017; Hüttermann, 2012). Based on this, we adopt previous attempts to define DevOps (Hemon-Hildgen, Rowe, et al., 2020) and combine them with AISD’s “ability to adapt to change” (Conboy, 2009) and the concept of “lean thinking” to holistically improve processes to continuously deliver valuable outcome to the customer (Fitzgerald and Stol, 2017; Poppendieck and Poppendieck, 2003). Our resulting conceptualization defines DevOps as “a set of continuously improved principles for collaborative work implemented between the IS development function and the IS operations function, along with potentially other stakeholders, which is founded on the sharing of culture, goals, measures, automation tools and automated processes towards continuous delivery of valuable outcome.” The latter aspect – continuously deliver value to the customer – provides a reason why DevOps in industry often bridges efficiency (e.g., through management and improvement of cycle time, Anderson and Reinertsen 2010; Wetherbe and Frolick 2000) and effectiveness (e.g., to deliver valuable outcome to the customer).

2.3 First Steps Towards Theorizing DevOps

Existing empirical research on DevOps so far has investigated various disparate aspects. Among those are, for example, the continuous integration of software development and the operation of the resulting IS product (Fitzgerald and Stol, 2017), the adoption challenges for DevOps (Lwakatare, Karvonen, et al., 2016), the necessary skill sets of team members for DevOps (Hemon-Hildgen, Lyonnet, et al., 2020; Wiedemann and Wiesche, 2018), or the control-alignment view for product orientation in DevOps teams (Wiedemann et al., 2019). These studies have also resulted in the proposition of two specific theoretical models.

First, a theoretical model of job satisfaction suggests that the orchestration of automation, sharing and risk management is moderated by work conditions and positively impacts job satisfaction if AISD teams move towards DevOps (Hemon-Hildgen, Rowe, et al., 2020). Second, a model on intra-IT alignment extends operational alignment’s focus on IT infrastructure and processes to alignment of development and operations functions (Wiedemann et al., 2020). Given mismatching interoperability, conflicting goals, different procedures, and various competencies between development and operations functions, the model proposes interrelated mechanics of integrated responsibility, individual componentization, and multidisciplinary knowledge in order to produce intra-IT alignment. Although these two models provide first very valuable preliminary insights on DevOps and some of its probable characteristics, mechanics, and effects, they offer only a narrow focus on very specific effects.

1 We subsume different developmental roles such as architect and developer into development, being aware that the industry-driven name of the DevOps phenomenon might be too exclusive regarding specific roles being involved in development and operations of the IS.
(job satisfaction) and mechanisms (intra-IT alignment), and are based on a single case study (a large service firm) and a very specific sample (smaller software products such as online shops) respectively. Important aspects that may lead to different DevOps implementations, characteristics, mechanisms, and effects are missing, for example, more complex IS products or larger organizational settings have been ignored so far. Existing studies have emphasized the importance of contextual factors while developing IS products (e.g., Baham and Hirschheim, 2021), and have highlighted the important role of contextual factors for transitions towards DevOps, without explicitly identifying and further examining them (for DevOps executions) (cf. Luz et al., 2019). Gaining understanding which contextual characteristics influence DevOps implementations and mechanisms considerably helps to unpuzzle the DevOps phenomenon. Considering the lack of research about DevOps in general, and the missing understanding which contextual characteristics exist that influence a specific DevOps implementation in particular, we explore the characteristics influencing current DevOps implementations. Our goal is to offer an empirically grounded understanding of factors that result in variations of DevOps implementations, and based on that, to abstract theory and to derive predictions for further DevOps implementations.

3 Research Method

3.1 Design Overview

We used an exploratory, inductive case study design (Eisenhardt, 1989). First, we identified appropriate case site candidates implementing DevOps. Previous research on DevOps also influenced the selection because we wanted to include similar but also contrasting case types to understand which characteristics led to these specific DevOps implementations.

Sampling criteria focused on revelatory cases (Dubé and Paré, 2003) in two groups of cases. The first group (group A) included case sites that are both revelatory and extreme while being successful in their domain, understanding IT as a core asset and an inherent driver for overall company success. These companies do run a holistic approach to ISD because business is driven by IT as a core asset for maximizing business value (Overby et al., 2006). These companies presumably are better integrated internally, and arguably do successfully soften conceptional barriers between development and operations. Case sites of the second group (group B) are comparable with members of the first group except that they run a more conservative, classic approach to IT. We expected them to apply DevOps moderately, thus providing a contrast.

We thoroughly examined pre-selected case site candidates that matched the sampling criteria and agreed to participate, and ran pilot interviews with 12 different firms over a period of ten months. The 12 firms were either part of our professional network, or were actively contacted for the research purpose at leading industry conferences. We started with one company from this set for each group, and iteratively added case sites to each case group. Since we gained meaningful insights with sufficient theoretical saturation (i.e., minimal incremental learning combined with minimal incremental improvement to the theory) after investigating four cases in total, we decided to end our data collection for this phase of the study.

The respective level of DevOps intensity (aligned with our DevOps definition) was the leading driver to understand the characteristics (leading to specific DevOps implementation) and to split the case groups, thus we did not highly prioritize the segments the firms are active in. However, we carefully paid attention to diverse sites across segments, leading to firms being in different businesses (i.e., online streaming, enterprise software, manufacturing, and financial industry in conjunction with automotive industry since the bank is a subsidiary of a leading car manufacturer), see Figure 1.
As part of this step we also determined the unit of analysis to understand the context of organizations tailoring their respective DevOps implementation. The unit of analysis is the respective enterprise. During the execution step, and cycling between the different steps of the execution stage, we recognized that even inside enterprises multiple different DevOps implementations may be in place, thus the unit of observation transitioned to teams of 10-30 persons who participate in development and delivery of IS products, including platform teams serving numerous other product teams.

### 3.2 Data Collection

Data was collected via interviews, including learning possible new questions from the interviews and snowball sampling to find out more knowledgeable informants (Eisenhardt, 1989). Complementary data sources included project documentations, public tech blogs and presentations, and, in parts, observation as a participant-as-observer (Myers, 2009; Yin, 2018). Aligned with the inductive, exploratory nature of this research, concepts and relationships were developed in sequential within-case analyses, before looking for similar concepts and relationships across multiple cases, to build and continuously further shape theory and to use theory to generalize from the case study.

We conducted 17 semi-structured interviews with highly knowledgeable informants from the case organizations spanning roles of engineers, architects, and decision makers. This allowed us for data triangulation within and across cases so that we could examine the phenomenon from different angles and gather as much alternative explanations as possible (Dubé and Paré, 2003). The open-ended interviews emphasized context and narratives (Brinkmann and Kvale, 2015; Mishler, 1986), were recorded, and lasted between 36 minutes and 107 minutes. We started our interviews with entry questions about the background of the informant, then discussed the transformation towards DevOps and DevOps dimensions asking for goals, practices, tools and any effect DevOps has on complexity, communication, or knowledge management. We closed interviews asking what next steps are planned in the team’s DevOps journey, asked for anything the informant wanted to add and for a recommendation for a colleague to talk to. Additionally, we triangulated data and addressed potential biases in several ways. First, one member of the research team is also an independent freelance consultant on DevOps. While this raises a potential bias, Cassell and Symon (2012) argue that some

\[\text{Participant observation was stopped due to the outbreak of the COVID-19 pandemic.}\]
degree of researcher bias is not only inevitable to the study, rather it is beneficial since such studies cannot be carried out in a social vacuum. Other argue the interviewer needs to be knowledgeable about the interview topic (Brinkmann and Kvale, 2015; Yin, 2018). Second, tactics to minimize participant bias included taking frequent breaks to continue on theory building, and to continuously reflect and analyse collected data. We also emphasized using multiple data sources and writing case writeups (Eisenhardt, 1989; Yin, 2018). In addition, we gave anonymity and transparency to our research participants, for example, by providing thorough information about the study.

### 3.3 Data Analysis

Following guidelines for building theory from case studies (Eisenhardt, 1989; Eisenhardt and Graebner, 2007), we analysed within-case and cross-case without any a priori hypotheses. The analysis was further influenced by the Grounded Theory Method (Strauss and Corbin, 2015). Particularly, we did not begin our analysis with any provisional codes, rather, we were guided by theoretical sampling following the flow the informants offered while asking exploratory questions. We always took special attention to remain open to alternative explanations, which helped us to understand the context-sensitivity of DevOps implementations. Our model emerged through continuously cycling between analysing the data and unfolding literature. We used the software MAXQDA for coding the transcribed interviews and additional documentation from the case study database. Data analysis followed established guidelines for coding practices for qualitative research (Saldaña, 2016; Strauss and Corbin, 2015). Specifically, as part of our first analysis cycle of initial coding, we broke down the rich data to discrete parts, and examined and compared them with other codes. Combined with pattern coding, we summarized segments of data and results and identified what we call “flavours of the continuum characteristics”, for example, continuous delivery. We then applied axial coding to identify the dominant codes and to explore theoretical relations. This led to categories that form our characteristics. Applied practices and success are example categories produced in this iterative step. Theoretical coding synthesized the categories derived from coding and analysis to create the emerging model. Example concepts that resulted from this step are inner environment and outer environment making up the core category environment, that in turn suggested that a specific DevOps implementation is highly context-sensitive. Figure 2 describes the coding process with illustrations.

![Figure 2. Coding process with illustrations.](image-url)
In addition, the analysis relied on memos, whiteboard sketching, tables, and graphs (Miles et al., 2020). While closing the cross-case analysis, as a blend of case evidence, prior research and stand-alone logic, we finalized the main findings as well as the model of the DevOps continuum (Eisenhardt and Graebner, 2007).

4 Findings

We assessed our cases to explore which characteristics impact how they implement DevOps. In the following we analytically summarize our findings and integrate the unveiled characteristics into a model. Our study reveals that the individual team aligns its DevOps initiative with characteristics of its inner environment and its outer environment, see Figure 3.

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Short descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Inner environment</strong></td>
<td></td>
</tr>
<tr>
<td>Organizational standards</td>
<td>Rules and norms the team must align with.</td>
</tr>
<tr>
<td>Dealing with change</td>
<td>How and when the team responds to changes.</td>
</tr>
<tr>
<td>Applied practices</td>
<td>Practices applied by the team.</td>
</tr>
<tr>
<td>Success</td>
<td>Success, and its measurement, the team aims to achieve to fulfill its task.</td>
</tr>
<tr>
<td><strong>Outer environment</strong></td>
<td></td>
</tr>
<tr>
<td>Market type</td>
<td>Attributes of the demand of the IS product the team provides, in its market.</td>
</tr>
<tr>
<td>Product type</td>
<td>Attributes and identifying features or core affordance of the IS product the team provides.</td>
</tr>
<tr>
<td>Digital technology</td>
<td>Digital technical means the team utilizes to develop and deliver the IS product.</td>
</tr>
</tbody>
</table>

Figure 3. Characteristics of teams’ DevOps implementations, with short descriptions.

The inner environment expresses characteristics having its root in the team or inherited from the organization it is embedded in. Although, from a team perspective, all other organizational entities in the same organization could be considered to be part of the outer environment, here, with outer environment, we express the context of the team to the field, its users and market. The inner environment and the outer environment make up the environment and summarize characteristics that influence how the team implements DevOps and thus determines the respective DevOps implementation. In sum, this leads not to a pre-defined set of configurations, but rather to a DevOps continuum, with gradual transitions of DevOps implementations in between two extremes, see Figure 4.

According to our DevOps definition, we define ‘no integration’ between two distinct functions development and operations as Detached Dev / Ops on the far left edge of the continuum, ‘full integration’ as Coalescence on the far right edge of the continuum, and Assisted DevOps as a ‘middle ground’, with dedicated IT functions of development and operations with strong collaboration, with
many nuances in between. Implementations of DevOps at SoftwareDev span the entire continuum, including classic enterprise resource planning (ERP) products on the left side and other teams with their respective context on the right side. MovieStream’s teams mostly can be positioned on the right side, not on the far right edge, since teams, although working strongly autonomously, utilize centralized platform teams. Platform teams are teams of highly specialized experts (e.g., security) providing an internal product or services (e.g., tools, infrastructure) for product teams (teams delivering IS products for a market as part of their outer environments). A context-sensitive variant of division of labour is formed where the whole team, with exception of highly specialized experts, is made up in a cross-functional way spanning different roles to develop and deliver the IS product. DevOps implementations of ManuFact, with its complex and heterogeneous industry portfolio, can be positioned along the entire continuum. The DevOps implementation of the examined team at AutoBank, with its specific context of two distinct functions of development and operations with only gentle collaboration, can be placed on the left side of the continuum.

4.1 The DevOps Continuum’s “Inner Environment”

We define characteristics of the inner environment as organizational standards, dealing with change, applied practices, and success. The first characteristic is the organizational standards. We define these standards to be corporate, normative rules the team must confirm to, the team setup, and the responsibilities of the team related to other organizational entities. Informants report that these standards contribute considerably to their working culture. Our data indicate that specific instances (i.e., flavours) of this characteristic vary. For example, employees of MovieStream only have to align with a few organizational requirements, leading to large degree of freedom of product teams to freely decide on how they accomplish their tasks. In contrast, teams at ManuFact have to align with more and stronger rules. For example, they must typically use centralized tools, provided and supported by other teams, for version management and binary management, to align with organizational compliance requirements. Different flavours also exist for division of labour across organizational entities. For example, while at MovieStream product teams also operate their product when it is running on production systems (“you build it, you run it”), they develop and freely share and switch operational activities across the product team. At AutoBank it is quite the opposite. There, due to a strict division of labour, colleagues from development do not have access to production systems and are not involved in operational concerns at all.

As regards the second characteristic, we define dealing with change as how and when the team responds to changes. Here as well different flavours exist, spanning a range from proactively seeking change and reactively avoiding any change at all. SoftwareDev pointed to situations where members of specific teams question changes and claim that they have solved a specific task the same way for 30 years. On the other side, at MovieStream, change is seen as an opportunity to learn and to understand customer needs. Continuously optimizing processes is mandatory for them if the team aims to deliver their product to production frequently.

The third characteristic as part of the inner environment, applied practices, concerns the methods the team uses to develop as well as deliver their IS product. Specific instances of applied practices span a broad range of flavours, from process-rich and manual, with long release cycles (e.g., at AutoBank, SoftwareDev), to continuous delivery (e.g., at MovieStream, SoftwareDev). Automation and sharing contribute to the concept of continuous delivery, where, in its extreme, software is delivered in a pace where the customer does not recognize that a new version of the IS product is in place at all. An important practice across all cases is to utilize automation particularly to foster collaboration and to streamline communication between development and operations. As a Product Owner from SoftwareDev succinctly puts it:

“Automating everything, and improving, is important to insure that it is running successfully, even if we know that we have gaps, but we need enough buffer to bring forward our product, in order to continue to make our customer happy”. (SoftwareDev, Product Owner)
Automation of handovers from development to operations reduces communication between involved parties. The entire IS delivery can be automated including compliance checks. The team reportedly feels to be much closer to the customer and being able to better react on their preferences, and thus improve customer satisfaction. In order to gather feedback faster, the team must be able to deliver faster, that is accomplished by automation.

Customer feedback is gathered continuously by implementing “observability”: Insights from user behavior and data in production are continuously gathered and metrics are shared across involved parties. As part of continuous delivery, “machine learning” supports to analyze issues on production systems, and various test-related activities are also executed on production systems including “experiment-driven development” (the behavior of the IS product on production systems is checked against defined expectations in form of hypotheses), “A/B tests” (the impact of the difference of two distinct production setups are compared with each other), and “Canary releases” (a new version of the IS product is made available on production systems to only a part of the user base). Across the cases, these practices are often combined and can lead to automatic rollbacks of production changes. A platform engineer of MovieStream gives an example.

“We have such high numbers of requests basically, you can learn a lot from metrics. Metrics is something extremely important. And any change being rolled out [to production], its metrics are monitored, basically to see if there’s any performance improvement or issues with it. The average change being rolled out is done with canary releases. There is a lot of automation about these things, and it’s very metrics-driven”. (MovieStream, Platform engineer #1)

The fourth and last characteristic is success (and its measurement), which we define as the measure the team aims to achieve to fulfill its task. For example, AutoBank has siloed success criteria where a development function focuses on completing many product features and the operations function focuses on stable production systems. In contrast, at MovieStream the approach to success is holistic and aligned with the experience of the user. They strongly measure how the customer uses the product and which business opportunities are possibly missed. Informants across all cases report efficiency as a business goal and DevOps as the enabler to get the speed to drive digital transformation.

4.2 The DevOps Continuum’s “Outer Environment”

We uncovered from our data that the outer environment of DevOps implementations includes the characteristics market type, product type, and digital technology. Market type subsumes the characteristic that defines attributes of the demand, that is the market pull. This may lead to a growing, fast changing business. The market type determines market conditions the team must face during their daily work. For example, in regulated markets and industries, it may be required by law or regulation to authorize changes in the company’s application systems, that staff must be trained before a new release or version of the IS can be put to production, that the team has to store all data of the application systems in data centers in a specific region, or that the IS product is to be hosted on production systems located in the customers’ technical network and must thus align with their compliance rules. A DevOps lead architect provides an example:

“Big companies ask for changes, but they might be stacked in order to check there on-site against local compliance tests, and then they are deployed to facility. Processes are different to ours, since we do not run our software”. (ManuFact, DevOps lead architect #3)

Unsurprisingly, we find that flavours of this characteristic differ widely between “consumer markets” and (online) business-to-customer products compared to “industry markets” and business-to-business products. For instance, to contrast to ManuFact, MovieStream is operating in a less-regulated consumer market; hence deploys to production multiple times a day, and can quickly rollback in case of any issues. The deployments strongly rely on automation.

We define the product type as the identifying feature or the core affordance of the provided IS product. For example, for MovieStream, the provided product allows users to consume streamed movies. In this case, the resulting IS product needs to be resilient and simple. Resilient means it acts robust
against failure situations and recovers from failures quickly. Simple means that MovieStream focuses on the feature that is directly available for the user. This very intense “product focus” stands in contrast to an enterprise focus for products, for example ERP systems used in enterprises. The product focus at MovieStream leads to few and simple use cases such as the ability to sign up for the service and the ability to press the play button.

The last characteristic of the outer environment is digital technology. We define this as the digital technical means to develop and deliver the IS product. Customers ask for new products and new features regularly, and nowadays the team can typically fulfill this demand without facing many technical limitations. This is the balance between “market pull” and “technology push”, as a DevOps lead architect from ManuFact explains:

“Software is used in many more industrial areas, that is the market pull, and on the other side we have the technology push, many data, and no restrictions such as memory, network bandwidth, or solutions in the cloud. The technology pushes. Digitization leads to many innovations and changes, I want to react on quickly. Agile software development is the development and DevOps includes the Ops part, and this helps the entire business to stay agile, to react on changes. If you don’t have this in the digitization age, ... it is like a life policy.” (ManuFact, DevOps lead architect #2)

The used digital technology needs to be managed. That is, although teams may have the autonomy and freedom to choose the technology they use to accomplish their work, they may use platforms and services, centralized in distinct organizational entities, to work efficiently and leverage their own resources on working on their IS product.

4.3 DevOps as a Funnel to Link Efficiency and Effectiveness

The data-based analysis of the teams’ environments and what is perceived as “IS success” offers an interesting lens on efficiency and effectiveness. On the one hand, DevOps is a means to develop and deliver a product in an efficient way, e.g. by continuously delivering new versions of the IS. On the other hand, DevOps bridges to effectiveness, because continuous delivery and automatically gathered insights derived from usage of the IS on production systems, support the teams to adjust, extend, or remove current IS functionality, or even find entirely new business opportunities:

“Actually, you do not want to ship features, you want to generate outcome for the users. If it is easy to deliver output, then you start to think about the value of this output.” (ManuFact, Dev manager)

Continuous delivery leads to a “shift left”, with development- and test-related activities running on the technical system the customer accesses to use the IS. The practice of continuous delivery spans involved IT functions of IS development and operations. It includes continuously probing the team’s environment to check whether the team should re-align based on given changes in its environment. Part of this is “sharing” across teams. Sharing emphasizes the context, as platform engineer of MovieStream points out while talking about the relationship to other autonomous teams, and his roles as a leader:

“I'm going to go over and talk to them, because I understand what they're trying to do, but understand how they're not going to get there. So, that context sharing is part of the responsibility of leaders within the organization to make sure that there's no information that stays so localized that other teams can't either leverage it or make use of it or understand what's going on. So, it actually required good communication and curation of information in leadership, so the teams can continue to operate in their locally specialized area, but not be disadvantaged. That's the socio part of our socio-technical system.” (MovieStream, Platform engineer #3)

Meanwhile, the team also must be able to deliver faster in order to gather feedback faster. Continuous delivery of this kind appears to get more difficult if the product type does not support that (e.g., if the IS product is not resilient or not enough meaningful metrics are available due to a low volume of user
requests) or the software is produced for and runs in industry markets (e.g., if the IS product is hosted at the customer and must confirm to their respective compliance rules, is shipped in an embedded way together with hardware, or many stakeholders are involved). A DevOps lead architect provides an example:

“We have this with our trains. If you deliver the IS product together with hardware, then it is getting difficult. Because then also the customers are specific, then you cannot just simply automate, because too many stakeholders are involved” (ManuFact, DevOps lead architect #1).

According to the perception of the studied cases, as evidenced by data indicating organizational standards, the more autonomously a team is, the faster and more flexible it can react based on gathered information and changing context. Our data analysis uncovered that with DevOps and high-performing teams, the fine-grained, autonomous team is an agent that can, based on simple rules, continuously adapt itself to fit to the changing inner and outer environment, and organically realign with its collaborative connections to other teams, while executives dynamically refit the architecture of teams, through adding, eliminating, combining and splitting teams. This happens under the permanent weighing the organizational balance between efficiency and effectiveness and probing new organizational standards. As a service manager from AutoBank states:

“Don’t just give the autonomy to this small unit, rather also enable the team. Give them the methodology technically and socially. [...] Create small units and empower them to decide themselves and support their collaboration with other capsules, to the other teams.” (AutoBank, Service manager)

Digital technologies, including those provided by other teams, can support the respective team to fulfil its tasks, but they can also thwart the team in case of restrictive organizational standards. If the team is empowered to decide on its best balance between efficiency and effectiveness, it steadily realigns with its inner environment. This brings the team closer to the edge of the right side of the continuum. For example, at MovieStream and SoftwareDev, teams may even completely bypass centralized platform products and services if they have good reasons to do so and can ensure that they still deliver aligned with operational requirements as part of organizational standards. From the team perspective, products and services provided by another team maybe considered to be “waste” in lean thinking and in this specific context, although, from an enterprise viewpoint, synergies may be missed. A product owner for tools from SoftDev emphasizes the continuous trade-off:

“If a team comes with new ideas, new languages, until the whole organization catches up with them, you need to find a way to support them. So, we invest into making easy what most teams are using. But on the other hand, we want to go in the direction of supporting what’s coming in. And that’s always a tradeoff, how much do you invest in that.” (SoftwareDev, Product owner tools #2)

Focusing too much on supportive activities that are already provided by other (platform) teams, may bind resources that could otherwise contribute to the creation of valuable outcome for the customer. In summary, empirical evidence suggests that DevOps is a vehicle to optimize the balance between efficiency and effectiveness.

5 Discussion

Our research aimed to shed light on which characteristics influence DevOps implementations. Similar to enterprises who customise AISD (Fitzgerald et al., 2006), enterprises tailor their respective DevOps implementations depending on their context. Surprisingly, we uncovered a broad set of characteristics, with typical flavours, each of them related to how DevOps is executed in a concrete context. What we observed is that even in one organization many different DevOps implementations may exist. The reason for this is, due to our analysis, that in one team, its environment may be very different compared to the environment of another team, both in one enterprise.
Our results show that variations of DevOps implementations are gradual transitions on a DevOps continuum between two edges, siloed IT functions of development and operations on the left, and full coalescence on the right edge. Coalescence practically means that an autonomous team (Lee and Xia, 2010) is fully responsible for developmental and operational concerns during IS development and owns the IS product (Cao et al., 2009).

Informed by the DevOps continuum, we unfold a streamlined role of success in DevOps initiatives. While implementing DevOps, goals and the definition of success evolve. Success can be conceptualized either from the developmental side (Siau et al., 2010), e.g. staying within scope, time, and cost requirements (Chow and Cao, 2008), functionality (Lee and Xia, 2010), rapid change (Lee and Xia, 2005), or delivering high quality (Siau et al., 2010), or from the operational side, e.g. the industry standard “mean time to restore” (Dekleva, 1992), MTTR in short, as the time required to restore the IS product after an incident occurs that makes the system unavailable. One of the most influential contributions (Urbach et al., 2009) focusing on the operational side of success is the IS Success Model (ISSM) (DeLone and McLean, 2016a; Petter et al., 2013).

The characteristic success relates to the reasons why DevOps is implemented at all. All cases worked with defined goals and strive for success. Cases articulated the strong importance of both instrumental as well as humanistic goals, indicating that instrumental goals are typically related to delivery performance (through continuous delivery) that is how fast and often an IS product can be made available to the user (Forsgren, 2018; Poppendieck and Poppendieck, 2003; Wetherbe and Frolick, 2000), a leading success criteria of DevOps implementations on the right side of the continuum. **Customer satisfaction** (DeLone and McLean, 2016b; Forsgren et al., 2016) can be considered to be a humanistic goal (Sarker et al., 2019). Thus, evidence exists, that customer satisfaction (Bhattacherjee, 2001; Fowler and Highsmith, 2001; Recker et al., 2017) becomes a leading success criteria, replacing more siloed measures for success, either from the developmental side or from the operational side, and emphasizing the concept of “user satisfaction” of the ISSM. Instrumental goals and humanistic goals are both holistic (spanning the IT functions of development and operations) and in its conjunction emphasize the sociotechnical nature of IS (Sarker et al., 2019). DevOps aims to reconcile the plethora of different approaches trying to measure IS success and ISD success.

Both, efficiency as well as effectiveness, can be a goal and thus define success. Traditionally, enterprises aim to optimize efficiency through cost synergies, e.g. to share given resources to serve a specific market, and single organizational entities, typically business units (BUs), aim to optimize their respective effectiveness. Those (with DevOps often fine-grained) units often decide which market to serve and find efficient ways to do so as long as the decision generally fits into the corporate strategy (Eisenhardt and Piezunka, 2011; Greenwood et al., 2020).

While DevOps at the first glance optimizes the efficiency of how teams develop and deliver the IS (the focus on delivery performance), it also optimizes effectiveness (the focus on customer satisfaction). Both concepts are intertwined. A team can perform efficiently while working not effectively (e.g., continuously delivering the wrong product) and vice versa (e.g., delivering the right product only once a year). High-performing teams strive for an optimal balance between efficiency and effectiveness. Continuously finding an optimal balance can only succeed if change is inherent ingredient of the process, “time-paced” evolution is in place and change is “learning-based” (Brown and Eisenhardt, 1997; Conboy, 2009), that in turn is a prerequisite to continuously deliver valuable IS products (Abrahamsson et al., 2009; Brown and Eisenhardt, 1997; Markus and Robey, 1988; Todnem By, 2005). With DevOps instances on the right side of the continuum, e.g. teams of SoftwareDev and MovieStream, platform teams aim to optimize efficiency while serving as an enabler for product teams with their respective foci on effectiveness.

In summary, DevOps is a funnel to link efficiency and effectiveness, for example, through insights gained from monitoring, fast feedback through continuous delivery, or continuously adapt to changes in its environment. Following this logic, DevOps is an enabler for digitization, it supports teams to concretize and continuously update the functional scope of the developed and delivered IS, and it includes teams permanently asserting their environments to deliver their “right outcome right”. The
study reveals that DevOps optimizes a balance between efficiency and effectiveness, since a team on the right side of the continuum continuously (“efficiently”) delivers valuable outcome to the customer (“effectiveness”) (Bosch, 2019).

6 Conclusion

This research adds significant and timely understanding of the emerging DevOps phenomenon to the academic field. First, it identifies characteristics that influence DevOps implementations. Characteristics, with their flavours, lead to DevOps implementations that can be positioned on a broad range that makes up the DevOps continuum. This model can support to explain existing and to predict future DevOps implementations. Second, this research contributes to existing concepts and theories. We uncovered the importance of “success” as a characteristic of the team’s inner environment, and the role of DevOps as a funnel to link efficiency and effectiveness. We are convinced that the results of this study help to further conceptualize DevOps and to unpuzzle the emerging DevOps phenomenon, and through the theoretical reasoning also contributes to understanding and theoretical grounding of the studied phenomenon. For practitioners, the suggested model can provide valuable guidance for initiatives towards DevOps. After we contributed to the basic understanding of DevOps and characteristics influencing implementations, future research might replicate this research to improve our confidence of generalizability of our findings. We call for studying other DevOps implementations, relate them to the continuum, and assess and expand the suggested model with its characteristics. We expect more possible lines of research about the DevOps phenomenon, once the theoretical grounding exists. We are convinced that our research is a valuable contribution into this direction, to gain understanding about DevOps as a logical evolution of AISD (that spans agile approaches across IT functions) and which characteristics influence their respective implementations.
References


