# Requirements Risk Management for Continuous Development: Organisational Needs

**Sanna Kainulainen**
Faculty of Information Technology, University of Jyvaskyla, Jyväskylän yliopisto, Finland

**Tuure Tuunanen***
tuure.t.tuunanen@jyu.fi, Faculty of Information Technology, University of Jyvaskyla, Jyväskylän yliopisto, Finland
tuure.t.tuunanen@jyu.fi

**Tero Vartiainen**
School of Technology and Innovations, Information Systems Science,
University of Vaasa, Vaasa, Finland

**Abstract**

Information systems development has recently evolved from traditional to agile and continuous forms. Continuous development (CD) methods, such as development and operations (DevOps), integrate many well-regarded parts of agile development and add collaboration among an organisation's development, operations and quality assessment departments. We argue that requirements risk management (RRM) poses additional challenges to projects where development work is carried out quickly and continuously. However, in the literature, most methods for prioritising requirements and managing risks are more suited to traditional development. This raises the need for new tools and methodologies to meet CD challenges. As these challenges constantly evolve, project management must be able to control CD, changes in the determination of requirements and the accompanying risks. Based on a systematic literature review, we define the key features of CD and develop a conceptual three-dimensional framework that can be used to understand the organisational needs of RRM for CD.

**Keywords***:* Requirements risk management, Agile, Continuous development, Development and operations, DevOps, Systematic literature review.

## 1 Introduction

Information systems (IS) development comprises the development activities required to create an IS. ISD involves multidisciplinary stakeholder collaboration to achieve better outcomes. Stakeholders influence the complex development process and are affected in turn (Maruping & Matook, 2020b; Siau et al., 2019). If cross-sectoral collaboration does not work, it can lead to higher requirement risks and project failures (Wang et al., 2016). Defining requirements has always been a critical part of project work. One of the most critical factors for moving a project forward and defining software functions and features is identifying industry-specific details and requirements. In agile projects, prioritising requirements and involving customers are critical to success (Stray et al., 2019).

Today's business constantly evolves; thus, continuous development (CD) requirements are increasing, and CD is seen as a future form of project development. Our study is interested in how requirements risk management (RRM) has been considered for CD, what specific characteristics have been defined for a CD project type and what tools and methods should be

Received 19 Mar 2023, accepted 14 May 2024 after 5 rounds of review with Karlheinz Kautz as accepting Editor

considered in risk management and definition. Previous studies have shown that using agile methods helps solve challenges with traditional methods and improves performance (Lwakatare et al., 2016). For example, the extreme programming method focuses on meeting customer requirements flexibly and interacting with customers during the development process to improve the provision and handling of continuous feedback. Adopting this method has been argued to require less reworking than traditional methods and, thus, be more cost-effective (Iyawa, 2020).

In contrast, CD focuses on organisational change because development is continuous and iterative, with no definite end. It aims to connect the development and operation domains to improve software development flow (Hemon-Hildgen et al., 2020), add services to the digital infrastructure and include continuous feedback and learning among actors from different organisational levels (Osmundsen & Bygstad, 2022). While organisations continue to move away from traditional stage gate approaches (Cois et al., 2014; Heemstra & Kusters, 1996; Royce, 1987) to agile and CD approaches (Hütterman, 2012), project management methods have not evolved at the same pace. Much of the literature posits that existing methods only suit traditional projects (Ghobadi & Mathiassen, 2016; Jiang et al., 2006; Racheva et al., 2010; Ramesh et al., 2010).

IS development requires new tools and techniques because traditional information systems development (ISD) methods and techniques are unsuitable for modern agile development (Kautz et al., 2007). Agile development enables a better understanding of customer needs and adaptation to today's needs (Cao & Ramesh, 2008; Kautz et al., 2007). Requirements usually come from business needs, and changing requirements threaten project success if not handled properly (Maruping et al., 2009). Maruping et al. (2009) argue that very little guidance is available on managing teams in agile development and that managing changing requirements as effectively as possible is essential because change is inevitable. Cao and Ramesh (2008) state that organisations need to be able to respond rapidly to merging change requests, which are critical and arise throughout the software development process. They differentiate the process of agile requirements engineering (RE) from traditional development, requiring new methods for handling requirements.

The above discussion raises important questions: If many methods and models are based on traditional development, when and where are requirements defined at the beginning of the project (Heemstra & Kusters, 1996)? More importantly, how should changing requirements be managed and prioritised with the latest post-agile CD approaches? We posit that CD requires methods focusing on change and feedback (Dingsøyr et al., 2019). In our study, we seek to answer the following research question: *How should requirement risks be managed in CD, and what are the organisational needs for accomplishing this?*

We apply a systematic literature review (SLR) method to identify CD's current state of the art and how it is defined and understood. Our analysis was conducted from both IS and RE perspectives.

Based on our SLR, we construct a framework with dimensions describing the organisational needs of RRM for CD. These dimensions can be used to define a project's aspects from a CD perspective and determine whether an organisation implements the critical CD development features and how the project meets the organisational needs of RRM for CD. The dimensions can be used to set preferences and priorities for project development to meet better CD requirements, such as rapid response and prioritisation of new needs.

This article is structured as follows. First, the theoretical background is presented, and the SLR process is described. The results are presented thereafter. The framework is then presented. Next, we discuss the findings and implications for research and practice. We conclude by discussing the study's limitations and offering avenues for future research.

## 2 Theoretical background

Wallace et al. (2004) define complexity risk and requirement risk as the technical dimensions of IS project risks: requirement risk describes the potential impact of a project's requirements or requirement management process. It represents the probability of project failure when an IS project's requirements are unclear or highly unstable.

Project requirement risk examples include missing or incorrect stakeholders, unclear, incomplete, inconsistent or unrealised requirements, undocumented or inaccurate assumptions, business requirements falsely defined as functional requirements, an inability to link applicable requirements to business requirements and unvalidated requirements. Projects based on flawed requirements will likely face challenges and problems and may fail (Venkatesh et al., 2018; Wallace et al., 2004).

In response to fast-changing requirements, agile development emerged as a new form of ISD to deliver new services (Mangalaraj et al., 2009; Mathiassen & Pries-Heje, 2006; Olszewska & Waldén, 2015). Agile methodology divides projects into phases, focusing on continuous collaboration and development. Teams follow a planning, implementation and evaluation cycle (Dingsøyr et al., 2019; Shimada et al., 2019).

CD, in turn, evolved from agile development. It extends agile development by focusing on short continuous learning and development cycles with a continuous feedback loop (Lwakatare et al., 2016; Osmundsen & Bygstad, 2022; Virmani, 2015).

Gatrell (2016) states, *"Technology has moved from continuous integration to continuous deployment and, finally, to continuous delivery"* (p. 104). Continuous integration combines several authors' source code changes into a single software project. Automated functions ensure the code's correctness and allow quick feedback on its quality (Gall & Pigni, 2021; Stray et al., 2019). Continuous deployment refers to frequent, automated software deployments (Gall & Pigni, 2021). Continuous delivery aims to shorten release cycles by automating software testing and acceptance (Chen, 2015; Ghantous & Gill, 2017). CD can be seen as an umbrella term that includes many DevOps processes, including continuous integration, testing, delivery and deployment (Osmundsen & Bygstad, 2022). According to Gall and Pigni (2021), advancing CD is critical for companies because agile methods cannot deliver quality results fast enough due to the market's dynamic nature.

In this article, development and operations (DevOps) are considered an instantiation of CD in which agile development is connected with fast delivery cycles, short feedback loops and automation. The CD is perceived as a way to organise functional units to achieve the necessary sensitivity and responsiveness to market conditions and demands (Maruping & Matook, 2020a; Wiedermann et al., 2020). DevOps is a CD approach that extends agile principles to the entire software process and cooperation between the operations, development and quality assessment domains, including testers and quality assurance teams (Ebert, 2018; Krey et al., 2022; Lwakatare et al., 2016; Osmundsen & Bygstad, 2022; Stray et al., 2019). DevOps can also be regarded as a set of new practices for deploying production changes more efficiently without compromising quality (Lwakatare et al., 2016; Ozkaya, 2019). Appendix 1 lists the key concepts and definitions used in this article.

Maruping and Matook (2020a) note that the academic literature has only just begun to understand the DevOps phenomenon and its impacts. Ghantous and Gill's (2017) literature review revealed no universal definition of DevOps. Despite the extensive literature on DevOps, it does not clearly explain what DevOps is and lacks conceptual inclusion. This lack is considered a significant barrier to the mainstream adoption of DevOps, preventing a thorough understanding of what DevOps includes and means and how organisations can successfully transition to DevOps (Gall & Pigni, 2021; Hüttermann & Rosenkrantz, 2019). For example, Krey et al. (2022) note that the implementation of DevOps in small and medium-sized enterprises has not been comprehensively researched.

## 3 Research Methodology: Systematic Literature Review

We chose an SLR for this study, applying Kitchenham et al.'s (2010) method to define the theoretical background. The SLR process is detailed in Appendices 2, 3 and 4. The literature review and analysis were used to create a conceptual framework, with each search step adding more detailed search criteria and further analysis. We examined 768 articles during the SLR, with 83 used to develop the framework.

First, we searched Google Scholar (separately for RRM and CD) for the Association for IS's Basket of Eight Journals[1] (AISBASKET8) with no time limit to examine how the IS literature addresses DevOps and CD in its top eight publications. Next, the first author conducted the first coding of the selected articles. This process began by the first author reading all the selected articles and highlighting text defining the features and concepts of continuous and agile development. Seventeen articles were transferred to ATLAS.ti, and several codes describing CD features or concepts were extracted from these articles.

Appendix 3 includes an example of the coding process. During the analysis, defining codes involved several systematic steps to ensure they accurately represented the key concepts and themes identified in the literature. The first step involved reading the articles and highlighting texts relevant to the research focus. These highlighted texts included vital phrases, sentences or paragraphs that address aspects of agile, DevOps, or Continuous Development (CD). The highlighted texts were then assigned initial codes by the first author. These initial codes were often descriptive labels that captured the essence of the highlighted text. For instance, a text discussing the importance of cooperation in DevOps might be initially coded as "cooperation".

After initial coding, the codes were reviewed and refined to ensure they accurately represented the highlighted texts. The authors reviewed the coding results together and worked on the code processing further in a workshop. This involved combining similar codes, splitting broad codes into more specific ones, and ensuring consistency in coding across different texts. For example, codes related to communication, cooperation, and knowledge

[1] *European Journal of Information Systems*, *Information Systems Journal*, *Information Systems Research*, *Journal of Association for Information Systems*, *Journal of Information Technology*, *Journal of Management Information Systems*, *Journal of Strategic Information Systems* and *MIS Quarterly*.

sharing were grouped into a code group related to project culture. Each code belonged to only one code group, ensuring that the classification was distinct and organised, and each code group represented a broader category of concepts. This involved articulating what each group encompassed and ensuring that all the codes within a group were relevant to this definition. The process was iterative, meaning that the first author would switch back and forth between the texts and the codes, continually refining and adjusting the codes and code groups. This ensured that the codes aligned with the article's content and the research objectives. Documentation was maintained throughout the coding process. This included justifying why certain texts were highlighted, specific codes were assigned, and code groups and dimensions were defined. This documentation was crucial for ensuring transparency and reproducibility of the analysis.

The final step of coding involved organising the code groups into more significant, overarching dimensions in a workshop. The all of the authors were involved in the process of creating the dimensions. As a result of this work, three dimensions were built: The first, culture, comprised subgroups – project and organisation. The other dimensions were methods and tools, and pace and seamlessness. These dimensions provided a structured lens for the analysis. Appendix 3, Table A4, lists the codes, code groups, and dimensions.

In summary, defining codes involved a systematic and iterative approach to highlighting relevant texts, assigning and refining initial codes, grouping similar codes into thematic code groups, and organising these groups into broader dimensions. This structured process ensured that the codes accurately reflected the key concepts and themes within the literature, providing a robust framework for analysis.

IS conference proceedings[2] were similarly handled, with 13 papers coded in ATLAS.ti. Next, articles found as part of the original search's forward and backward reference search results (other than the AISBASKET8 journal articles and IS conference proceedings papers) were selected for coding.

The SLR's first phase focused on IS, encoding the concepts IS research focused on in DevOps and CD. We next wanted to determine how the IS literature studies DevOps and CD and which concepts it focuses on in terms of the earlier identified framework dimensions. Articles were selected similarly to the first search, resulting in 25 papers analysed using the defined research lens with three new dimensions. The purpose was to discover how key features of agile development, DevOps and CD are described and how the features of the different dimensions are highlighted. Appendix 2, Tables A1 and A2 list the search results and number of selected articles, papers and conference proceedings.

The initial results revealed a lack of focus on CD: 16 articles (64%) focused on methods and tools, 12 (48%) on culture/project culture structure, 7 (28%) on culture/organisational culture structure and 4 (16%) on development speed or seamlessness. Most articles mentioned only traditional development methods; 28% mentioned agile development, but only 4% mentioned CD.

[2] *The International Conference on Applied Mathematics, Informatics, and Computing Software (AMICS), European Conference on Information Systems (ECIS), Hawaii International Conference on System Sciences (HICSS), International Conference on Information Systems (ICIS)* and *Pacific Asia Conference on Information Systems (PACIS).*

Next, we expanded the search to cover the corresponding literature in a selection of RE journals[3]. Three searches were performed separately for DevOps and CD, the last of which resulted in 159 articles, with 8 selected for analysis. Appendix 2, Table A3, lists the search results and selected articles. The articles were again coded with ATLAS.ti. The coding results were very similar to those for the IS articles, with similar concepts emerging. This reinforced our results from the IS article analysis and provided further evidence to refine the framework. Five (63%) articles focused on methods and tools, 3 (38%) on culture/project culture, 4 (50%) on culture/organisational culture and 5 (63%) on development speed or seamlessness. Most articles commented on agile, DevOps or CD.

Appendix 4 lists all analysed and referenced articles. Our descriptive analyses of the IS and RE literature show an almost equal focus on the dimensions of methods and tools: 64% of the IS articles and 63% of the RE articles. The difference was slightly higher for the culture/project culture dimension: 48% of the IS articles and 38% of the RE articles. The most significant differences were in the culture/organisational culture and development speed or seamlessness dimensions. The IS articles focused much less on these dimensions than the RE articles. Only 28% of the IS articles concentrated on elements of the culture/organisational culture dimension; in the RE articles, the coverage was 50%. The difference was even more significant in the development speed or seamlessness dimension. Only 16% of the IS articles presented this dimension's elements, while in the RE articles, the coverage was 63%. There was also a clear difference in development style mentions. In the IS literature, only traditional methods were usually mentioned, and there was little focus on CD. In contrast, in the RE literature, most articles mentioned agile development, DevOps or CD.

## 4 Findings

Based on our SLR, we identify the key features and concepts that can be used to determine the characteristics of CD methods. Table 1 summarises the three dimensions of CD: culture, methods and tools and pace and seamlessness.

Culture is divided into project culture and organisational culture. Project culture describes how a project's organisation is managed, its functionality, knowledge sharing, roles and groups. Because information is fragmented within an organisation, projects must involve people with in-depth knowledge of the development objectives and risks. Everyone should share information and knowledge openly, as changes occur quickly and development work is continuous.

Organisational culture focuses on the organisation doing the development work. The critical features are cooperation, work tasks and overall transformation of the organisational culture based on CD. While progress in CD is achieved in collaboration with the firm's development, operations and quality assurance domains, users and customers are also actively involved in the work. This work aims to develop a purpose-built, flexible system tailored to customers' and users' needs and requirements. Development work must quickly and seamlessly progress to meet new challenges, requirements and risks. Testing and development operate together, and the project must have the right resources to guide the development work in the correct

[3] *Empirical Software Engineering: An International Journal, IEEE Software, IEEE Transactions on Software Engineering, Information and Software Technology, Information Systems (IS), Requirements Engineering Journal, Software and Systems Modelling, Software Practice and Expertise* and *Software Quality Journal*.

direction. CD and constant change require action to keep the organisation responsive. Information technology (IT) projects are particularly vulnerable to commitment challenges, mainly due to requirements volatility and software's intangible nature (Horlach et al., 2020; Lee et al., 2021).

The literature shows there is a demand for new methods and tools to keep project management consistent with CD (see, e.g. Babb et al., 2017; Bragge & Merisalo-Rantanen, 2009; Hüttermann & Rosenkranz, 2019; Lwakatare et al., 2016).

The methods and tools dimension focuses on the need for practical and usable tools to implement risk management in CD and the need to develop valuable and cost-saving tools for handling risk management. The pace and seamlessness dimension refers to the ability to respond quickly to requirements and change requests, to work in an iterative CD environment and the possibility to receive and react to continuous feedback from customers and users.

| | |
|---|---|
| Dimension 1a: Culture – Project Culture | |
| Definition | Organisation's approach to project management and implementation. |
| Activities | Development, operations, knowledge sharing, cooperation, defining roles and groups and continuing cooperation with users and customers. |
| Explanation | Project work is built on seamless cooperation and knowledge sharing among developers, users, operations staff and customers; requirements and different groups and users define risks, and practitioners with the proper knowledge must participate in the project. The focus is on continuous feedback and self-organisation. |
| Dimension 1b: Culture – Organisational Culture | |
| Definition | Organisation's approach to managing CD and its further development. |
| Activities | Development, operations and quality assurance cooperation. |
| Explanation | Development organisation includes developers, operations and quality assurance teams. Development is not only done by developers. Cooperation among different departments is essential. |
| Dimension 2: Methods and tools | |
| Definition | The means to support change cycles and fast development according to changing requirements. |
| Activities | Adoption and development of methods and tools suitable for CD. |
| Explanation | Methods must support rapid, cyclical development and changing requirements and risks. Tools must account for cultural effects and user values when eliciting requirements. Knowledge-sharing tools are essential because development is based on knowledge-sharing. |
| Dimension 3: Pace and seamlessness | |
| Definition | The development process is rapid, continuous and without interruptions. |
| Activities | Rapid development, continuous work and continuous, flexible processes. |
| Explanation | Development is a continuous, flexible process that can adapt to changing demands and adjust to constant feedback. |

*Table 1. Dimensions of organisational needs of RRM for CD.*

In the following subsections, the three dimensions are discussed in more detail, including perspectives from the RE and RRM literature.

### 4.1 Project culture for RRM

This dimension relates to the development and operations of a project and to its users and customers for the results. When defining a project's organisation, the project team members' expertise is paramount. While collaboration and knowledge sharing often identify broad risks (Jiang et al., 2006), it is essential to focus on changing requirements and project goals (Maruping et al., 2009). Requirements ambiguity contributes to software project challenges in critical domains. Failure is less likely when the right people are involved (Niederman et al., 1991). However, unclear and changing requirements in the middle of a project are among the significant challenges of software development projects. As a result, it is almost impossible for development teams to identify and meet all customer expectations (Ghanbari, 2016).

Similarly, an inability to respond to changing user requirements is one of the most critical reasons for project failure. Failure also includes delayed project schedules, budget overruns and poor quality (Maruping et al., 2009; Mathiassen et al., 2007). These factors also contribute to the difficulty of sharing knowledge between individuals. Previous research has shown that organisational culture influences how knowledge is transferred and stored and affects the success of managing IT project requirements (Azizi & Rowlands, 2018).

Identifying the roles and key stakeholders in a project is an essential first step in risk management. Li et al. (2003) state that the IS user environment needs more research, while Keil et al. (2002) explore how different roles can help identify project risks. People in different positions will recognise diverse project risks. Groups can also define risks differently, so it is vital to maintain open communication and have good knowledge-sharing practices and discussion opportunities (Keil et al., 2002; Li et al., 2003). It is well recognised that managing risks in IT projects is essential and that failure to address them might cause project problems, such as user dissatisfaction (Elbanna & Sarker, 2015; Keil et al., 2002; Li et al., 2003; Ramesh et al., 2010). A necessary part of project success is communication between users and developers. Although merging different stakeholders' knowledge is recognised as a critical part of project success, there is no research on integrating different stakeholders into project risk management (Keil et al., 2002; Li et al., 2003).

Project risks typically relate to users (Elbanna & Sarker, 2015) and include poor communication with users and stakeholders at the project level, a lack of user involvement or misunderstanding of user requirements or failure to manage user expectations or accommodate defined requirements and scope changes. Communication among different groups and roles critically influences risk management (Li et al., 2003; Ramesh et al., 2010). However, an organisation's ability to define risks strongly affects a project's quality (Karlsson et al., 2007). It is also essential to introduce development methods to users, incorporate them into the project and coordinate expertise so that specialised knowledge is spread and integrated among the project's many roles or phases (Gemino et al., 2007; Patnayakuni et al., 2006). This can be perceived as a practical risk management approach (Gemino et al., 2007).

According to the agile manifesto, collaboration and responsiveness to change are essential. Several studies (e.g. Azizi & Rowlands, 2018; Ghanbari, 2016; Kiper, 2016; Sletholt et al., 2012; Stray et al., 2019) identify lack of knowledge about requirements and organised activities and testing principles as problem areas. Identifying and defining requirements are problematic because they may be predefined or unknown (Sletholt et al., 2012). Therefore, the requirement planning phase should be extended with activities that collect and exploit new sources of information (i.e. development activities should be more closely integrated with operational activities). The possibility of using direct user feedback and a central infrastructure also poses challenges for testing and validation processes (Stuckenberg & Heinzl, 2010). Once software requirements have been received from stakeholders, they should be validated to ensure they meet user needs. Requirements must also be prioritised to address technical constraints, business considerations and critical stakeholder preferences (Kiper, 2016).

Babb et al. (2017) assert that a lack of knowledge sharing is a problem for agile development and CD. Similarly, Ghobadi and Mathiassen (2017) observe that agile development lacks appropriate tools to manage risks associated with knowledge sharing and introduce a theoretical model for mitigating such risks in several ways to assess and clarify project information-sharing risk profiles and aiming to create an overall plan for reducing and resolving risks. Their study's results highlight how different risk management profiles for information sharing can lead to varying project performance outcomes. Their model introduces concepts and detailed processes for managing a project's knowledge-sharing risks. Davison (2017) remarks that with this model, it is easy to show how risk management can be used in agile development to achieve better performance and outcomes.

In examining how progressive obstacles to knowledge sharing are observed and differ according to the observer's role, Ghobadi and Mathiassen (2016) conclude that sharing information is challenging and influenced by the actors' knowledge of the work organisation and environment. In turn, organisational success depends on how effectively employees share information (Qureshi et al., 2018). Knowledgeable practitioners must actively participate in the project (Taylor et al., 2012). It is typically assumed that managers are well informed about possible risks in using project methods, but this is not always so (Schmidt et al., 2001). However, such knowledge is often fragmented throughout the organisation, so project managers should employ experienced users who can share information and work collaboratively (Tiwana & Keil, 2004). Barriers can be caused by internal organisational tensions, value systems, personality clashes and policies related to knowledge transfer.

### 4.2 Organisational culture for RRM

The organisational culture dimension relates to changes in organisational culture and cooperation among different departments. In many organisations, managing risks is one of the most critical challenges. Taylor et al. (2012) note that organisations often do not apply research or knowledge on risk management and risk factors. This is a problem, considering that the project outcome depends on how well the organisation prioritises its requirements (Karlsson et al., 2007). Up to a third of development costs are due to incorrect requirements (Patnayakuni et al., 2006). Jiang et al. (2006) describe partnering as a possible solution for improving RRM in that cooperation between users and IT staff creates new possibilities to define requirements better. Collaborative knowledge exchange positively impacts the whole development process and performance.

The importance of cooperation is stressed in agile development methods, which require collaboration between development teams and customers, including customer feedback, throughout the project (Cao et al., 2009; Xiao et al., 2018). The literature identifies the importance of a designated customer representative role in an agile project (Maruping &

Matook, 2020b; Matook & Maruping, 2014). However, managing this role can be challenging, as it is multifaceted and takes many forms. Mapping the modes of action the role requires is not easy because hardly any theory exists on the definitions of the different modes of action (Maruping & Matook, 2020b).

The DevOps team comprises staff from the development, operations and quality assessment departments. Their collaboration is critical, and changing the organisation's structure is important for these previously independent departments to become part of one development unit. Requirements definitions and related risks pose additional challenges to ongoing development projects in which development is quick and continuous. While traditional development requirements are defined at the project's beginning, agile development requirements are defined iteratively throughout the project (Ramesh et al., 2010). The same applies to CD projects. Krancher et al. (2018) identify continuous feedback and self-organisation as other key issues in CD. Rapid feedback and other DevOps practices help create and strengthen the development team's autonomy (Callanan & Spillane, 2016) and focus on valuable features and requirements (Chen, 2015).

The lack of both knowledge and guidance is challenging for organisations shifting to DevOps (Gall & Pigni, 2021). Previous studies have shown that agile development methodologies focus on ISD (Dev) but pay little attention to the operational (Ops) aspects of software deployment in a production environment (Gall & Pigni, 2021; Ghantous & Gill, 2017). The definition and concepts of DevOps are still unclear, and there is little understanding of the factors that influence the adoption of DevOps practices in an organisation (Gall & Pigni, 2021; Ghantous & Gill, 2017). Sharp and Babb (2018) highlight the fact that there is no universally accepted definition of DevOps and note fundamental differences between the Software Engineering and IS literature; in IS, the conceptual elements of DevOps remain undefined. This lack of a commonly agreed-upon definition of DevOps is a critical area for future research (Sharp & Bagg, 2018).

Traditionally, different organisational units, teams and individuals coordinate. In agile contexts, organisational structure changes are encouraged, limiting the stability and alignment they can provide. Therefore, alignment must allow flexible structural design, while letting people work together as smoothly as possible. From a customer perspective, agility is the ability to continue delivering customer value (Horlach et al., 2020). Ramesh et al. (2010) state that a customer's inability to provide the correct requirements to the development team and a lack of harmony between developers and customers significantly impact the development process, for example, if requirements are poorly drafted. Chen (2015) comments that despite the extant literature on organisational change, little research focuses on introducing continuous delivery or development to an organisation.

Conversely, according to Lwakatare et al. (2019), studies of successful adoptions and implementations of CD, specifically DevOps, show organisations' abilities to change their structures, processes and tools. Osmundsen and Bygstad (2022) recognise the potential of such changes in organisations, enabling them to respond to customer needs and requirements more innovatively. User input and communication across different organisational levels are considered the leading factors enabling value delivery in CD. Cao et al. (2009) note that the literature recommends using agile methods in organisations with a flat structure. In organisations with centralised and hierarchical structures, the organisational culture may cause problems between the top management and the project team.

## 4.3 RRM methods and tools

Different means to support developers to track requirements are needed because requirements constantly evolve. Requirements must be continuously modified and improved during a project, necessitating methods and tools to help developers track requirements. However, such advanced tools in smaller projects may be considered unnecessary (Ghanbari, 2016).

Davis (1982) emphasised the importance of determining requirements and observed obstacles to and challenges in defining correct requirements: human limitations and complexity of and variety in information and user–analyst interaction models. Various methods and tools are needed to overcome these obstacles and challenges. Heemstra and Kusters (1996) present different tools and methods for risk management elicitation and definition in various steps and project phases, determining that the purpose of risk prioritisation in traditional projects is to choose from identified risks. The most important should be on a manageable list at the project's beginning.

Tiwana and Keil's (2004) one-minute risk assessment tool aims to help project teams conduct "what-if" analyses and improve software practices. It comprises project risk levels and questions to estimate an overall risk score. Project managers and stakeholders assign scores to each question, and the results produce a background image of each project's risk exposure. They argue that the most critical risk driver is the choice of methodology, followed by customer involvement.

Taylor et al. (2012) note inconsistent methods, and Ramesh et al. (2010) report a lack of information on requirements management in real agile projects. Several checklists for defining risks have been developed, but few organisations adopt them (Wallace et al., 2004). The challenge lies in creating practical tools for implementing risk management.

As there is no consistent model for managing knowledge-sharing risks in agile projects, Ghobadi and Mathiassen's (2017) tool aims to help understand and manage knowledge-sharing risks in agile development environments when moving from project risks to effective knowledge-sharing and resolution strategy plans. It presents seven risk area categories and five resolution strategies. For example, one defined risk area is team diversity, referring to conceptual, geographical and time differences between team members that may hinder effective knowledge sharing. The resolution strategy to overcome this is strengthening resources (i.e. strategies to develop supportive capabilities, experiences and technologies).

Ghobadi and Mathiassen (2017) note that agile development must find a way to identify customers' requirements and demands. In turn, Tuunanen and Kuo (2015) argue for the need for tools that account for cultural influences and user values. Similarly, Bragge and Merisalo-Rantanen (2009) state that challenges remain involving users in development, especially with traditional methodologies. Simply put, traditional methods do not include explicit information about when and in which part of the project user involvement should occur.

## 4.4 Pace and seamlessness for RRM

In CD, development is rapid, continuous and seamless. This is based on business demands, where quick responses to changes and new requirements are mandatory. Organisations must be able to deliver agile IS, whereby development phases are repeated multiple times in cycles and requirements are defined and fulfilled at the beginning of each iteration cycle (Hickey & Davis, 2004; Patnayakuni et al., 2006). While several different agile approaches exist, all focus on development that can respond flexibly and seamlessly to customers' and developers' requests and needs (Elbanna & Sarker, 2015).

According to Ramesh et al. (2010), traditional requirement methods and practices are often too cumbersome to evolve with the rapidly changing field of agile development. They propose new techniques to help execute the RE process in agile projects to compensate for the limitations of traditional methods. An example is in-person communication instead of written specifications to transfer ideas between customers and the development team without extensive documentation and generate a flexible way to consider various requirements (Xiao et al., 2018).

Another good technique to define requirements in agile development involves user stories that can be used to determine high-level requirements (Cao & Ramesh, 2008). Elbanna and Sarker (2015) note that short iteration cycles allow changes to be considered faster and more efficiently, and user requirements can be clarified more effectively. Face-to-face interaction and ongoing communication with users and business groups improve this relationship and help develop a common understanding of project requirements. Racheva et al. (2010) state that the agile RE literature has too little information on how reprioritisation work is done in practice, and generic models of how they should be processed are missing.

Chen et al. (2016) comment that feedback loops must be open in agile development. Feedback should include technical and business feedback, such as performance, security and availability issues and new requirements, such as new user features. Handling feedback should be continuous, a learning cycle and part of the development cycle. Krancher et al. (2018) note that continuous frequent feedback is essential for gathering customers' ideas and requirements; it should be handled as fast as possible, and learning cycles should be kept as short as possible to be able to learn as much as possible.

## 5 Discussion

Digitalisation and today's business demands create challenges requiring new ways of working (Ebert, 2018). We argue that CD is a new way to develop IS systems to meet rapidly changing situations and requirements (Ramesh et al., 2010). Consequently, we determined the literature's position on CD and its instantiation of DevOps. We determined key dimensions of CD: (1a) culture – project culture, (1b) culture – organisational culture, (2) methods and tools and (3) pace and seamlessness. More specifically, we explored how these development methods accommodate RRM needs to answer our research question.

Our findings align with other research. Gall and Pigni (2021) define a continuous culture concept that corresponds to the project and organisational culture dimensions; continuous monitoring corresponds to the methods and tools dimension; and continuous automation, to some extent, matches the pace and seamlessness dimension. However, notably our findings show that the IS literature does not focus on automation (continuous automation, monitoring, etc.) but mostly on other aspects. Therefore, the dimensions defined in this study cannot be directly compared with Gall and Pigni's (2021) model. Krey et al. (2022) argue that DevOps implementations consist of the following interrelated categories: agility, collaboration, automation, measurement, monitoring and transparency. These categories can be used to find correspondences between our three dimensions and the elements of Gall and Pigni's (2021) model.

Jayakody and Wijayanayake (2023) group the critical success factors of DevOps into four areas: collaborative culture, DevOps practices, skilled DevOps teams and metrics and measurement. They argue that, in addition to culture, DevOps affects the processes, products, related technologies and organisational structures used in software development and operations processes. There is a clear convergence between these concepts and the dimensions and key concepts defined in our research. Jha et al. (2023) state three key themes associated with DevOps culture and thinking: collaboration, continuous improvement and automation, corresponding with the dimensions defined in our study. Finally, Khan et al. (2022) state that *"culture, practices, and tools are the three backbones of DevOps; culture defines a way of thinking with some basic standards. Practices reflect culture's significant success, and numerous tools are required to implement these methods"* (p. 14339).

Our findings also reveal that the extant IS and RE literature has, to some extent, focused on traditional-style waterfall model development and the applicable tools and methods (see, e.g. Ghobadi & Mathiassen, 2016; Highsmith & Cockburn, 2009; Jiang et al., 2006; Racheva et al., 2010). In the traditional model, at the beginning of a project, the objectives, requirements and schedules are defined (Heemstra & Kusters, 1996) and implemented by a specific group of experts and project management professionals in the IT department. Cao and Ramesh (2008) echo this finding by arguing that there is a scarcity of research on how RE is managed in agile projects. Ramesh et al. (2010) state that the literature does not elaborate well on how RE activities are considered in agile projects. This is problematic, as CD projects require seamless cooperation from developers, users and operations staff. The pace of change in CD is faster than that of legacy agile development (Highsmith & Cockburn, 2009), and many tools are designed for slower development work. This highlights the importance of knowledge sharing and cooperation, which have always been crucial in development projects but are even more so with CD.

## 5.1 Implications for research and practice and a roadmap

We believe that CD requires new methods and tools to modernise the requirements definition process and improve knowledge sharing without disrupting development work (Ghobadi & Mathiassen, 2017; Ramesh et al., 2010). It is also essential that development considers changing requirements throughout the process. However, we argue that this can create problems with knowledge sharing, as different stakeholder groups often have other priorities (Ghobadi & Mathiassen, 2016), which can result in a further lack of cooperation (Li et al., 2003; Ramesh et al., 2010).

Our findings highlight the need to change project organisation and structures to involve the necessary stakeholders with appropriate knowledge (Gemino et al., 2007; Patnayakuni et al., 2006; Taylor et al., 2012). Additionally, knowledge sharing and internal communication require development and a new way of thinking collaboratively and creatively. For example, brainstorming or frequent social interaction through IT can effectively share information (Babb et al., 2017; Qureshi et al., 2018). Organisations need to collaborate and create structures that support CD work. Essential components are iterative requirements, definition work and self-organisation (Cao et al., 2009; Krancher et al., 2018; Ramesh et al., 2010). Our research also shows the need to introduce CD to an organisation and simultaneously reformat its culture and structure to support a cyclical, continuous way of working (Cao et al., 2009; Chen, 2015; Matook & Maruping, 2014; Xiao et al., 2018).

The developed framework highlights the cultural changes required to enable an organisation to work in accordance with CD. It helps the organisation prepare for the challenges of CD, improve collaboration between different departments, and structure the project organisation to work with CD to achieve seamless collaboration and knowledge sharing between all stakeholders, including ongoing collaboration with users and customers. The CD also requires the IT department to be restructured so that there is a seamless collaboration between development, operations, and QA rather than departments working in isolation. The CD also requires new methods and tools to support a continuous, rapid development cycle, respond to continuous feedback, encourage and improve knowledge sharing, and consider cultural influences and user values as part of the requirements definition.

Agile methodologies are inherently different from CD requirements in terms of challenges and characteristics: continuous feedback, interdepartmental cooperation, collaboration, changing organisational structure, rapid and seamless continuous development, forming a project organisation. Our framework emphasises organisational and project culture changes, collaboration, cooperation, feedback loops and seamless automation, which also means new tools. With new methods and tools that support continuous, rapid development cycles, respond to continuous feedback, encourage and improve knowledge sharing, and help to consider cultural influences and user values when defining requirements.

When we compare Maruping et al. (2009)'s work with ours, we note that our framework highlights more the need for feedback loops and advanced monitoring, adapting development strategies based on real-time data and changing user requirements. Furthermore, our framework enables organisations to further improve their ability to effectively manage changing user requirements, optimise development processes, and deliver high-quality software products in dynamic and rapidly evolving environments.

Ramesh et al. (2010), in turn, developed agile RE practices in their study. We argue that, to support CD, these practices need to incorporate DevOps and CD principles such as automation, collaboration, and continuous delivery into agile practices, as well as a feedback mechanism such as implementing feedback loops, as our framework emphasises. Thus, RE practices must support continuous feedback throughout the CD lifecycle - collecting, analysing and acting on feedback from stakeholders, users and team members. In line with our framework, real-time collaboration and communication are essential in CD and require new tools tailored to CD methodologies.

In the following, we present a roadmap for CD RRM based on the dimensions of the developed framework and propose some research problems and related research questions. We also suggest research approaches, objectives, and, finally, how managers would need to address the risks associated with CD requirements. Table 2 summarises this roadmap.

### 5.1.1 Project Culture: Stakeholder Roles and Knowledge Sharing

First of all, we find that the CD project comprises various stakeholders from different parts of the business or organisation. It includes multiple roles and areas of expertise from developers, operations staff, users and customers. Customer engagement is essential in digital transformation and development (Sebastian et al., 2017). Communication among different stakeholders is critical for a project's success (Keil et al., 2002; Li et al., 2003; Ramesh et al., 2010). Knowledge sharing is part of organisational success and depends on how effectively stakeholders share information (Qureshi et al., 2018). It is an essential part of project work, as is seamless cooperation among the project stakeholders, including users and customers (Dingsøyr et al., 2019; Elbanna & Sarker, 2015; Krancher et al., 2018; Tiwana & Keil, 2004). Different stakeholders also define requirements and risks, which can be modified/added to as part of the development cycle based on their feedback during the project. In CD, practitioners with appropriate knowledge must participate in the project. Project and development work focus on continuous feedback and self-organisation (Krancher et al., 2018; Matook & Maruping, 2014).

According to our study, managers need to address requirement risks in CD projects by perceiving them as business projects, not merely as IT projects. Stakeholders and knowledge can be found in various departments, roles, and users. Managers should focus on changing requirements, project goals, and identified risks. They must prioritise requirements to address technical constraints, business considerations, and critical stakeholder preferences. It is essential to introduce practical risk management approaches and tools to users and integrate them into the project while coordinating expertise. Information should be shared through open communication and good knowledge-sharing practices.

We suggest an in-depth qualitative study (Klein & Myers, 1999) of CD teams to investigate project culture, focusing on roles and knowledge-sharing dynamics. This approach analyses real-world CD team structures to determine their effectiveness in managing requirement risks.

#### 5.1.2 Organisational Culture: Structure and Cross-Department Collaboration

Consequently, the organisational structure of CD brings a significant change in organisational culture compared with traditional development (Dremel et al., 2017; Mathiassen & Pries-Heje, 2006). In CD (and DevOps), the organisational structure is built on cooperation among different departments, including development, operations and quality assurance. Collaboration among various departments is essential (Dremel et al., 2017; Lwakatare et al., 2019; Ozkaya, 2019). However, our findings show that an organisation's hierarchical structure prevents cooperation across borders and complicates development work (Dremel et al., 2017), creating poor communication between stakeholders and easily creating sporadic development work (Lwakatare et al., 2019). We argue that businesses must play a substantial role in development work and ensure that their needs and requests are considered part of the project and that they obtain what the business needs (Matook & Maruping, 2014).

To effectively support CD, the project organisation must find a way to involve all necessary stakeholders in both the project and project risk management. They need to create an environment where knowledge sharing among different organisational departments, users, and stakeholders is developed and prioritised. Managers should create a cohesive structure that allows previously independent departments to function as a unified development unit, facilitating cooperation and ensuring that the needs and requests of all stakeholders are included and addressed throughout the project lifecycle.

Furthermore, businesses need to assign a substantial role to development work and ensure that their needs and requests are included in the project and that they obtain what they need. Managers should change the organisation's structure so that previously independent departments become part of one development unit. Cooperation between users and IT staff should be enabled to define requirements iteratively throughout the project and prioritise them. Continuous definition and configuration work, feedback, and self-organisation should focus on the most valuable features and requirements. This comprehensive approach ensures that CD projects are effective and aligned with business needs, fostering a collaborative environment that can adapt to changing requirements and risks.

To support CD, organisational culture and structure must be developed to promote collaboration, flexibility, and transparency. This involves creating a unified development unit where previously siloed departments work together seamlessly, facilitating communication and cooperation. An organisational culture that encourages iterative feedback and values continuous improvement is essential. Such a culture supports the iterative definition and prioritisation of requirements, ensuring that development work remains aligned with business goals.

Organisational knowledge plays a crucial role in the success of RRM. Businesses can more accurately identify and articulate their needs by leveraging the organisation's collective expertise and insights. This shared knowledge base helps in the iterative requirement definition process, making it easier to adjust to changes and effectively incorporate feedback. Ensuring that all stakeholders are well-informed and engaged in the development process enhances the overall quality and relevance of the requirements.

Establishing robust feedback mechanisms and maintaining close collaboration between IT staff and business users is vital to ensuring the accuracy of the business's requirements. Regularly scheduled reviews, continuous configuration work, and self-organised teams can help capture the most critical and valuable features.

Additionally, a mixed methods study (Venkatesh et al., 2013) examining cultural factors within CD environments and their effects on feedback processes is recommended. The objective of such a study is to investigate how organisational culture influences the implementation of continuous feedback loops in CD. Understanding these cultural dynamics can provide insights into improving the accuracy and effectiveness of requirements management, ultimately contributing to the success of CD projects.

### 5.1.3 Methods and Tools: Innovation for Rapid Cyclical Development

Our findings also reveal that most tools and methods have been created for traditional development work (e.g. Ghobadi & Mathiassen, 2016; Highsmith & Cockburn, 2009; Jiang et al., 2006; Racheva et al., 2010; Tuunanen & Kuo, 2015) and do not consider the functionalities required for CD. Consequently, new methods that support rapid cyclical development are needed. These tools must handle changing requirements and risks and account for users' cultural influences and values when eliciting requirements (Tuunanen & Kuo, 2015). CD highlights the importance of knowledge-sharing tools because development work is based firmly on knowledge sharing (Ghobadi & Mathiassen, 2017). This point of view has neither been part of traditional development nor considered when designing tools for traditional project management (Bragge & Merisalo-Rantanen, 2009).

Managers, therefore, should address requirement risks in CD projects by treating the choice of methodology and customer involvement as the most critical risk drivers of the project. They should introduce new tools and methods to respond to cyclical development processes and handle changing requirements and risks, including cultural effects and user values. Managers should also focus on the changing requirements, project goals, and identified risks. They must prioritise requirements to address technical constraints, business considerations, and critical stakeholder preferences. Additionally, they should introduce practical risk management approaches and tools to users, integrating them into the project while coordinating expertise. Effective information sharing through open communication and good knowledge-sharing practices is essential.

We suggest conducting design science research (Hevner et al., 2004; Peffers et al., 2007) studies to assess the utility and performance of tools and their impact on risk management at a team level. A goal of such a study should be to evaluate the effectiveness of new tools and techniques for managing dynamic requirements in CD projects, thereby contributing to the advancement of CD practices.

Reflecting on these points, several critical questions emerge. How can customer requirements and demands be identified in CD? This involves engaging customers early and continuously through iterative feedback loops, using user stories, personas, and direct communication to understand and refine their needs. How can users be involved in the development process, and how should user involvement occur? Users should be integrated into the development process through regular feedback sessions, usability testing, and collaborative workshops. Their involvement should be structured to ensure continuous input and validation of the product's direction and functionality. Addressing these questions is essential for refining methodologies and ensuring the successful implementation of CD practices.

#### 5.1.4 Pace and Seamlessness: Rapid Iteration

We also find that the pace and seamlessness dimension differs between traditional development and CD. Traditional development is typically a rigid, pre-planned process whose results are available when all the development work has been completed, and the project closed, whereas CD’s development process is fast, continuous and seamless. The pace of change in CD is even quicker than in agile development (Highsmith & Cockburn, 2009). CD can flexibly adapt to changing demands and adjust to continuous feedback. Changes are implemented as fast as possible; cycles can vary from many times a day to a few times a week or month (Chen, 2015). The key argument is that users and customers do not need to wait long to receive replies to their new requests or feedback, and the short iteration cycles and frequent releases help reduce risks, clarify requirements and increase the speed of feedback (Elbanna & Sarker, 2015; Lwakatare et al., 2019). Consequently, CD’s objective is to implement system changes as a cyclical process with no definite end.

Managers should, thus, focus on short iteration cycles and frequent releases when addressing requirement risks in CD. To achieve this, managers should develop and implement open feedback loops and learning cycles that are as fast and short as possible. They should enable continuous customer and user feedback that is processed quickly and seamlessly from the customer’s or user’s point of view, ensuring that ideas and requirements are gathered effectively.

To effectively implement CD, the organisational workflow needs to be restructured to support rapid development cycles and manage changing requirements. Cross-departmental communication must be enhanced to ensure all teams are aligned and respond swiftly to changes. This might involve integrating cross-functional teams and ensuring that development, operations, quality assurance, and other relevant departments work closely together from the start. Such integration helps in quicker identification and resolution of issues. Automating processes, especially testing, integration, and deployment, is crucial for speeding up development cycles and reducing human error. Continuous training and development are also essential to keep teams updated with CD's latest tools and practices, maintaining high efficiency and adaptability.

The project must establish robust mechanisms to provide quick responses and manage risks associated with change requests and new requirements. Real-time monitoring and feedback systems are vital, providing real-time insights into the system's performance and continuously gathering user feedback. Developing frameworks that assess the risks of each change request based on impact, urgency, and feasibility helps prioritise tasks and manage resources efficiently. Adopting an incremental update approach, where changes are broken down into smaller, manageable updates, minimises the risk of significant disruptions and makes identifying and addressing issues easier. Engaging stakeholders actively through regular updates and reviews ensures their needs are understood and met promptly, aligning project goals with business requirements.

A multi-case study (see, e.g., Stake, 2013) of CD projects focusing on workflow changes and their impact on risk reduction can be carried out to explore and develop the dimension of pace and seamlessness. The aim is to examine workflow changes that enable CD teams to respond quickly to change requests while managing the risks associated with requirements. This study could analyse different CD implementations to compare how various organisations have modified their workflows to accommodate CD and the outcomes of these changes.

Evaluating risk management practices and documenting best practices for integrating feedback loops, automating processes, and fostering cross-departmental collaboration would provide valuable insights. By reflecting on these questions and incorporating the findings into their strategies, managers can better navigate the challenges of CD and leverage its benefits for faster, more responsive, and less risky software development.

## 6 Conclusions

In this research, we summarised the IS and RE literature on RRM for CD to understand the state of the art, revealing that there is still little focus on RRM for CD. Through an SLR, we constructed a framework with dimensions describing the organisational needs of RRM for CD. This conceptual framework can set preferences and priorities for project development to better meet CD requirements, such as rapid response and prioritisation of emerging needs.

We posit that CD and RRM should proceed in an interleaved step-by-step manner, with the development cycle considering new requirements and their associated risks as part of a continuous improvement process. This approach requires a change in project culture and an organisation's operations and culture. Above all, it requires new methods and tools so that the integration of RRM into the CD process is a fast, seamless and practical part of the development process. We also presented research topics and questions that should be considered in future studies (Table 2).

| CD element(s) | Implications for Research and Suggested Research Questions | Suggested Research Approaches and Objectives | Implications for Practice: How managers need to address requirement risks in CD? |
|---|---|---|---|
| Project Culture | A change in project organisation is needed when moving to CD. This puts forward research questions such as the following:<br>How should project organisation be structured to support CD?<br>How can all the necessary stakeholders be involved in project and project risk management?<br>How can knowledge sharing among different organisational departments, users and stakeholders be developed?<br>What approaches and tools are needed in a project to manage risks associated with knowledge sharing and requirements?<br>Focus on managing the organisation's internal communications and operating environment. | Approach: In-depth qualitative study of CD teams, focusing on roles and knowledge-sharing dynamics.<br>Objective: Analyze real-world CD team structures to determine their effectiveness in managing requirement risks. | Projects should be perceived as business projects, not only as IT projects. Stakeholders and knowledge can be found in many different departments, roles, users, etc. Managers should therefore:<br>Focus on the changing requirements, project goals and identified risks.<br>Prioritise requirements to address technical constraints, business considerations and critical stakeholder preferences.<br>Introduce practical risk management approaches and tools to users, and integrate them into the project and coordinates expertise.<br>Share information through open communication and good knowledge-sharing practices. |
| Organisational Culture | Changes need to be made so that the organisational structure can support continuous feedback, definitions of iterative requirements and self-organisation in CD. This puts forward research questions such as the following:<br>How do organisational culture and structure need to be developed to support CD?<br>How can organisational knowledge contribute to the success of RRM?<br>How can the accuracy of the requirements defined by the business be ensured? | Approach: Mixed methods study examining cultural factors within CD environments and their effects on feedback processes.<br>Objective: Investigate how organisational culture influences the implementation of continuous feedback loops in CD. | Businesses need to assign a substantial role to development work and ensure that their needs and requests are included in the project and that they obtain what they need. Managers should therefore:<br>Change the organisation's structure so that previously independent departments become part of one development unit.<br>Enable cooperation between users and IT staff to define requirements iteratively throughout the project and prioritise them. |

| | | | |
|---|---|---|---|
| | | | Address continuous definition and configuration work, feedback and self-organisation focusing on most valuable features and requirements. |
| Methods and Tools | New methods that support fast, cyclical development are needed. These tools should be able to handle changing requirements and risks and consider cultural effects and user values when eliciting requirements. This puts forward research questions such as the following:<br>How can customer requirements and demands be identified in CD?<br>How can users be involved in the development process, and how should user involvement occur? | Approach: Design science research study conducted with CD teams, assessing tool utility and performance and its impact on risk management.<br>Objective: Evaluate the utility and effectiveness of new tools and techniques for managing dynamic requirements in CD projects. | The choice of methodology and customer involvement need to be treated as the most critical risk drivers of the project. Managers should therefore:<br>Introduce new tools and methods to respond to cyclical development processes and handle changing requirements and risks, including cultural effects and user values. |
| Pace and Seamlessness | Pace and change in CD are faster than in agile development and CD should be studied from multiple perspectives, such as well-being at work. This puts forward research questions such as the following:<br>How does the organisational workflow need to be changed to communicate across different departments and work with fast-developing cycles and changing requirements?<br>How can the project provide quick responses and define risks to change requests and new requirements defined by the business? | Approach: A multiple case study of the CD projects, focusing on workflow modifications and their impact on risk mitigation.<br>Objective: Examine real-world workflow adaptations that enable CD teams to respond quickly to change requests while managing requirement risks. | In CD, focusing on short iteration cycles and frequent releases is essential. Managers should therefore:<br>Develop and implement open feedback loops and learning cycles as fast and short as possible.<br>Enable continuous customer/user feedback that is processed quickly and seamlessly from a customer's or user's point of view to gather ideas and requirements. |

*Table 2. Roadmap for RRM research and practice in CD*

Our study has some limitations. The AISBASKET8, conference proceedings and a few other sources cover only part of the IS field. We applied a rigorous SLR approach, but this study is not a comprehensive literature review. We aimed to investigate how mainstream IS journals address CD and RRM issues. Similarly, we reviewed top journals publishing RE research to obtain their perspectives. In addition, our research question and choice of keyword searches contributed to the limitations of the SLR methodology. These limitations may also be reflected in the reduction of data and data extraction accuracy. Another related limitation is related to the use of Google Scholar to search the literature. Use of other literature sources, such ones offered by the academic publishers (e.g., EBSCO or Proquest), may potentially result different search results depending on the algorythms used by different search engine providers. In addition, the mapping exercise used systematic reading of the articles to determine the keyword annotation. Thus, intrepretive assessment was used in the data analysis and coding, which may have introduced bias. We also recognize that our study could be extended by searching the Senior Scholars' List of Premier Journals and choosing articles from a field of Software Engineering other than RE. In addition, our search used specific selected terms; it can be considered whether these terms were sufficient and appropriate.

Application of the framework has some boundary conditions. Based on the SLR, the framework's dimensions are the key challenges faced by a traditional development organisation when adopting RRM for CD. Consequently, our framework can be used to 1) define the state of the organisation and development work in terms of the CD definitions, 2) how the organisation's and project culture have evolved and 3) how they compare with the characteristics defined in the framework. In addition, we can use the framework to examine, for example, how the tools and methods used in an organisation work for CD development, what changes are needed to support CD, how knowledge sharing works in the organisation and how effective collaboration is. Therefore, our framework can be used to determine an organisation's capacity and status to adopt or use CD. It can also outline where changes should be made in an organisation to make the rapid cyclical improvement model work without unnecessary interruptions, delays or knowledge-sharing problems.

In our ongoing study, we seek to answer the question of how changing requirements should be managed and prioritised in CD and further develop an RRM method we have been working on (Tuunanen et al., 2023). For this, we apply a design science research approach (Hevner et al., 2004; Peffers et al., 2007) and the developed framework. We are testing the RRM method in an industrial environment and using it in a multinational enterprise resource planning system implementation and deployment project. Our initial goal is to determine how to further develop the RRM method to better address CD projects' needs and mitigate practitioners' concerns about a new tool for managing project requirements risks. We see that the developed conceptual framework can help prioritise requirements and determine their scope and severity and thus directly impact our RRM method development. The framework may help determine how agile methodologies and their associated requirements affect team performance (Maruping et al., 2009), which may have implications for our study. In addition, we are considering using Ramesh et al.'s (2010) framework to assess the applicability of the method and identify requirements risk practices to further improve the method. It would also be interesting to see if our conceptual framework can assist us in defining and managing cultural change in organisations and how this, in turn, may impact our ongoing study.

To conclude, according to Almeida et al. (2022), DevOps applications lack clear implementation guidelines for organisations. Since the most critical challenge identified in DevOps is cultural change (Jayakody & Wijayanayake, 2023)—specifically, the change in collaboration culture—we anticipate that the developed framework can offer ways to tackle this challenge in the development method in general, but also specifically for RRM for CD. Khan et al. (2022) also state that DevOps needs further research on which methods to adopt and how to apply and improve them, as DevOps requires learning new tools, skills and social norms. This is something we believe as well and our design science research study should offer insights to resolving these challenges, in addition to developing a novel method.

## References

Almeida, F., Simões, J., & Lopes, S. (2022). Exploring the benefits of combining devops and agile. *Future Internet*, *14*(2), 63.

Azizi, N., & Rowlands, B. (2018). The moderating effects of organisational culture on the relationship between knowledge sharing and IT risk management success. In *Proceedings of the 26th European Conference on Information Systems (ECIS)*, 1–10.

Babb, J. S., Nørbjerg, J., & Yates, D. J. (2017). The empire strikes back: The end of agile as we know it? In H. Holone, S. Koch Stiberg, & J. Karlsen (Eds.), *Selected papers of the IRIS; issue 8*, 44–49. IRIS.

Baham, C., & Hirschheim, R. (2022). Issues, challenges, and a proposed theoretical core of agile software development research. *Information Systems Journal*, *32*(1), 103–129.

Bragge, J., & Merisalo-Rantanen, H. (2009). Engineering e-collaboration processes to obtain innovative end-user feedback on advanced web-based information systems. *Journal of the Association for Information Systems*, *10*(3), 196–220.

Callanan, M., & Spillane, A. (2016). DevOps: Making it easy to do the right thing. *IEEE Software*, *33*(3), 53–59.

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.

Cao, L., & Ramesh, B. (2008). Agile requirements engineering practices: An empirical study. *IEEE Software*, *25*(1), 60–67.

Chen, L. (2015). Continuous delivery: Huge benefits, but challenges too. *IEEE Software*, *32*(2), 50–54.

Chen, H.-M., Kazman, R., & Haziyev, S. (2016). Agile big data analytics development: An architecture-centric approach. In T. X. Bui & R. H. Sprague, Jr. (Eds.) *Proceedings of the 49th Hawaii International Conference on System Sciences (HICSS)*, 5378–5387, IEEE.

Cois, C. A., Yankel, J., & Connell, A. (2014). Modern DevOps: Optimizing software development through effective system interactions. In *Proceedings of the 2014 IEEE International Professional Communication Conference (IPCC)*, 1–7, IEEE.

Davern, M., Shaft, T., & Te'eni, D. (2012). More enduring questions in cognitive IS research: A reply. *Journal of the Association for Information Systems*, *13*(12), 1012–1016.

Davis, G. B. (1982). Strategies for information requirements determination. *IBM Systems Journal*, *21*(1), 4–30.

Davison, R. M. (2017). Editorial: The limitations of limitations. *Information Systems Journal, 27*(6), 695–697.

Dingsøyr, T., Falessi, D., & Power, K. (2019). Agile development at scale: The next frontier. *IEEE Software, 36*(2), 30–38.

Dremel, C., Wulf, J., Herterich, M. M., Waizmann, J. C., & Brenner, W. (2017). How AUDI AG established big data analytics in its digital transformation. *MIS Quarterly Executive, 16*(2), 81–100.

Ebert, C. (2018). 50 years of software engineering: Progress and perils. *IEEE Software, 35*(5), 94–101.

Elbanna, A., & Sarker, S. (2015). The risks of agile software development: Learning from adopters. *IEEE Software, 33*(5), 72–79.

Gall, M., & Pigni, F. (2021). Taking DevOps mainstream: A critical review and conceptual framework. *European Journal of Information Systems, 31*(5), 548–567.

Gantman, S. (2011). Boundary objects and internal control in outsourced ISD projects: Results of a pilot study. In *Proceedings of the 17th Americas Conference on Information Systems (AMCIS)* (pp. 1–9).

Gatrell, M. (2016). The value of a single solution for end-to-end ALM tool support. *IEEE Software, 33*(5), 103–105.

Gemino, A., Reich, B. H., & Sauer, C. (2007). A temporal model of information technology project performance. *Journal of Management Information Systems, 24*(3), 9–44.

Ghanbari, H. (2016). Seeking technical debt in critical software development projects: An exploratory field study. In *Proceedings of the 49th Hawaii International Conference on System Sciences (HICSS)* (pp. 5407–5416). IEEE.

Ghantous, G. B., & Gill, A. (2017). DevOps: Concepts, practices, tools, benefits and challenges. In *Proceedings of the Pacific Asia Conference on Information Systems (PACIS)* (pp. 1–12).

Ghobadi, S., & Mathiassen, L. (2016). Perceived barriers to effective knowledge sharing in agile software teams. *Information Systems Journal, 26*(2), 95–125.

Ghobadi, S., & Mathiassen, L. (2017). Risks to effective knowledge sharing in agile software teams: A model for assessing and mitigating risks. *Information Systems Journal, 27*(6), 699–731.

Heemstra, F. J., & Kusters, R. J. (1996). Dealing with risk: A practical approach. *Journal of Information Technology, 11*(4), 333–346.

Hemon-Hildgen, A., Rowe, F., & Monnier-Senicourt, L. (2020). Orchestrating automation and sharing in DevOps teams: A revelatory case of job satisfaction factors, risk and work conditions. *European Journal of Information Systems, 29*(5), 474–499.

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS Quarterly, 28*(1), 75–105.

Hickey, A. M., & Davis, A. M. (2004). A unified model of requirements elicitation. *Journal of Management Information Systems, 20*(4), 65–84.

Highsmith, J., & Cockburn, A. (2009). Agile software development: The business of innovation. *Computer*, *34*(9), 120–127.

Horlach, B., Drews, P., Drechsler, A., Schirmer, I., & Böhmann, T. (2020). Reconceptualising business-IT alignment for enabling organisational agility. In *Proceedings of the 28th European Conference on Information Systems (ECIS), An Online AIS Conference, June 15–17, 2020.*

Hütterman, M. (2012). *DevOps for developers*. Apress.

Hüttermann, M., & Rosenkranz, C. (2019). DevOps: Walking the shadowy bridge from development success to information systems success. *In Proceedings of the 40th International Conference on Information Systems (ICIS),* 1–9.

Iyawa, G. E. (2020). Personal extreme programming: Exploring developers' adoption. In *Proceedings of the 26th Americas Conference on Information Systems (AMCIS)*, 1–10.

Jayakody, V., & Wijayanayake, J. (2023). Critical success factors for DevOps adoption in information systems development. *International Journal of Information Systems and Project Management*, *11*(3), 60–82.

Jha, A. V., Teri, R., Verma, S., Tarafder, S., Bhowmik, W., Kumar Mishra, S., Appasani, B., Srinivasulu, A., & Philibert, N. (2023). From theory to practice: Understanding DevOps culture and mindset. *Cogent Engineering, 10*(1), 1–31.

Jiang, J. J., Klein, G., & Chen, H. G. (2006). The effects of user partnering and user non-support on project performance. *Journal of the Association for Information Systems*, *7*(2), 68–90.

Karlsson, L., Thelin, T., Regnell, B., Berander, P., & Wohlin, C. (2007). Pair-wise comparisons versus planning game partitioning – experiments on requirements prioritization techniques. *Empirical Software Engineering*, *12*(1), 3–33.

Kautz, K., Madsen, S., and Nørbjerg, J. (2007). Persistent problems and practices in information systems development. *Information Systems Journal*, *17*(3), 217–239.

Keil, M., Tiwana, A., & Bush, A. (2002). Reconciling user and project manager perceptions of IT project risk: A Delphi study. *Information Systems Journal*, *12*(2), 103–119.

Khan, M. S., Khan, A. W., Khan, F., Khan, M. A., & Whangbo, T. K. (2022). Critical challenges to adopt DevOps culture in software organizations: A systematic review. *IEEE Access*, *10*, 14339–14349.

Kiper, J. R. (2016). Needs to know: Validating user needs for a proposed FBI Academy Knowledge Management System. In *Proceedings of the 49th Hawaii International Conference on System Sciences (HICSS)* (pp. 4334–4343). IEEE.

Kitchenham, B., Pretorius, R., Budgen, D., Brereton, O. P., Turner, M., Niazi, M., & Linkman, S. (2010). Systematic literature reviews in software engineering—A tertiary study. *Information and Software Technology*, *52*(8), 792–805.

Klein, H. K., & Myers, M. D. (1999). A set of principles for conducting and evaluating interpretive field studies in information systems. *MIS quarterly*, *23*(1), 67-93.

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.

Krey, M., Kabbout, A., Osmani, L., & Saliji, A. (2022). DevOps adoption: Challenges & barriers. In *Proceedings of the 55th Hawaii International Conference on System Sciences (HICSS), virtual, January 3–7, 2022*, 7297–7309, University of Hawai'i at Manoa.

Lee, J. S., Keil, M., & Wong, K. F. E. (2021). When a growth mindset can backfire and cause escalation of commitment to a troubled information technology project. *Information Systems Journal*, *31*(1), 7–32.

Li, E. Y., Jiang, J. J., & Klein, G. (2003). The impact of organizational coordination and climate on marketing executives' satisfaction with information systems services. *Journal of the Association for Information Systems*, *4*(1), 99–117.

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? In *Proceedings of the 49th Hawaii International Conference on System Sciences (HICSS)*, 5437–5446. IEEE.

Lwakatare, L. E., Kilamo, T., Karvonen, T., Sauvola, T., Heikkilä, V., Itkonen, J., Kuvaja, P., Mikkonen, T., Oivo, M., & Lassenius, C. (2019). DevOps in practice: A multiple case study of five companies. *Information and Software Technology*, *114*(1), 217–230.

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.

Maruping, L. M., & Matook, S. (2020a). The evolution of software development orchestration: Current state and an agenda for future research. *European Journal of Information Systems*, *29*(5), 443–457.

Maruping, L. M., & Matook, S. (2020b). The multiplex nature of the customer representative role in agile information systems development. *MIS Quarterly*, *44*(3), 1411–1437.

Maruping, L. M., Venkatesh, V., & Agarwal, R. (2009). A control theory perspective on agile methodology use and changing user requirements. *Information Systems Research*, *20*(3), 377–399.

Mathiassen, L., & Pries-Heje, J. (2006). Business agility and diffusion of information technology. *European Journal of Information Systems*, *15*(2), 116–119.

Mathiassen, L., Saarinen, T., Tuunanen, T., & Rossi, M. (2007). A contingency model for requirements development. *Journal of Association of Information Systems*, *8*(11), 569–597.

Matook, S., & Maruping, L. M. (2014). A competency model for customer representatives in agile software development projects. *MIS Quarterly Executive*, *13*(2), 77–95.

Niederman, F., Brancheau, J. C., & Wetherbe, J. C. (1991). Information systems management issues for the 1990s. *MIS Quarterly*, *15*(4), 475–500.

Olszewska, M., & Waldén, M. (2015). DevOps meets formal modelling in high-criticality complex systems. In *Proceedings of the 1st International Workshop on Quality-Aware DevOps (QUDOS)*, 7–12. ACM Press.

Osmundsen, K., & Bygstad, B. (2022). Making sense of continuous development of digital infrastructures. *Journal of Information Technology*, *37*(2), 144–164.

Ozkaya, I. (2019). Are DevOps and automation our next silver bullet? *IEEE Software*, *36*(4), 3–5.

Patnayakuni, R., Ruppel, C. P., & Rai, A. (2006). Managing the complementarity of knowledge integration and process formalization for systems development performance. *Journal of the Association for Information Systems*, *7*(8), 545–567.

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of Management Information Systems*, *24*(3), 45–77.

Qureshi, I., Fang, Y., Haggerty, N., Compeau, D. R., & Zhang, X. (2018). IT-mediated social interactions and knowledge sharing: Role of competence-based trust and background heterogeneity. *Information Systems Journal*, *28*(5), 929–955.

Racheva, Z., Daneva, M., Herrmann, A., & Wieringa, R. J. (2010). A conceptual model and process for client-driven agile requirements prioritization. In P. Loucopoulos & J. L. Cavarero (Eds.), *Proceedings of the Research Challenges in Information Science (RCIS), 2010 4th International Conference*, 287–298, IEEE.

Ramesh, B., Cao, L., & Baskerville, R. (2010). Agile requirements engineering practices and challenges: An empirical study. *Information Systems Journal*, *20*(5), 449–480.

Rowe, F. (2014). What literature review is not: Diversity, boundaries and recommendations. *European Journal of Information Systems*, *23*(3), 241–255.

Royce, W. W. (1987). Managing the development of large software systems: Concepts and techniques. In *Proceedings of the 9th International Conference of Software Engineering (ICSE '87)*, 328–338, ACM Press.

Salmela, H., Baiyere, A., Tapanainen, T., & Galliers, R. D. (2022). Digital agility: Conceptualizing agility for the digital era. *Journal of the Association for Information Systems*, *23*(5), 1080–1101.

Schmidt, R., Lyytinen, K., Keil, M., & Cule, P. (2001). Identifying software project risks: An international Delphi study. *Journal of Management Information Systems*, *17*(4), 5–36.

Sebastian, I. M., Ross, J. W., Beath, C., Mocker, M., Moloney, K. G., & Fonstad, N. O. (2017). How big old companies navigate digital transformation. *MIS Quarterly Executive*, *16*(3), 197–213.

Sharp, J., & Babb, J. (2018). Is information systems late to the party? The current state of DevOps research in the association for information systems eLibrary. In *Proceedings of the 24th Americas Conference on Information Systems (AMCIS)*,1–8.

Shimada, T., Ang Soo-Keng, J., & Ee, D. (2019). Exploring the impact of IS function maturity and IS planning process on IS planning success: An ACE analysis. *European Journal of Information Systems*, *28*(4), 457–472.

Siau, K., Long, Y., & Ling, M. (2010). Toward a unified model of information systems development success. *Journal of Database Management*, *21*(1), 80–101.

Sletholt, M. T., Hannay, J. E., Pfahl, D., & Langtangen, H. P. (2012). What do we know about scientific software development's agile practices? *Computing in Science & Engineering*, 14(2), 24–37.

Stake, R. E. (2013). *Multiple case study analysis*. Guilford press.

Stray, V., Moe, N. B., & Aasheim, A. (2019). Dependency management in large-scale agile: A case study of DevOps teams. In *Proceedings of the 52nd Hawaii International Conference on System Sciences (HICSS*), 7007–7016.

Stuckenberg, S., & Heinzl, A. (2010). The impact of the software-as-a-service concept on the underlying software and service development processes. In *Proceedings of the Pacific Asia Conference on Information Systems (PACIS)*,1297–1308.

Taylor, H., Artman, E., & Woelfer, J. P. (2012). Information technology project risk management: Bridging the gap between research and practice. *Journal of Information Technology*, *27*(1), 17–34.

Tiwana, A., & Keil, M. (2004). The one-minute risk assessment tool. *Communications of the ACM*, *47*(11), 73–77.

Tuunanen, T., & Kuo, I. T. (2015). The effect of culture on requirements: A value-based view of prioritization. *European Journal of Information Systems*, *24*(3), 295–313.

Tuunanen, T., Vartiainen, T., Kainulainen, S., & Ebrahim, M. (2023). Development of an Agile Requirements Risk Prioritization Method: A Design Science Research Study. *Communications of the Association for Information Systems*, *52*(1), 609-637.

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.

Venkatesh, V., Rai, A., & Maruping, L. M. (2018). Information systems projects and individual developer outcomes: Role of project managers and process control. *Information Systems Research*, *29*(1), 127–148.

Virmani, M. (2015). Understanding DevOps & bridging the gap from continuous integration to continuous delivery. In *Proceedings of the 5th International Conference on the Innovative Computing Technology (INTECH 2015)*, 78–82, IEEE.

Wallace, L., Keil, M., & Rai, A. (2004). How software project risk affects project performance: An investigation of the dimensions of risk and an exploratory model. *Decision Sciences*, *35*(2), 289–321.

Wang, S. Y., Chang, T. H., Hsu, J. S. C., & Lin, T. C. (2016). A study of the influences of knowledge boundary spanning on project performance in information system development projects. In *Proceedings of the Pacific Asia Conference on Information Systems (PACIS)* (pp. 1–10).

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*, *29*(5), 458–473.

Xiao, X., Lindberg, A., Hansen, S., & Lyytinen, K. (2018). 'Computing' requirements for open source software: A distributed cognitive approach. *Journal of the Association for Information Systems*, *19*(12), 1217–1252.

## Appendix 1. Key concepts and definitions

| | |
|---|---|
| Information Systems Development (ISD) | Development activities that are required to create an information system. (Kautz et al., 2007; Gantman, 2011) |
| Agile Development | Methodology focusing on collaboration, efficiency, and flexibility in software development. (Cao & Ramesh, 2008; Dingsøyr et al., 2019; Maruping et al., 2009; Mathiassen & Pries-Heje, 2006;) |
| Continuous Development (CD) | An umbrella term that includes many DevOps processes, including continuous integration, testing, delivery, and deployment.<br>Extends agile development by focusing on continuous and short learning cycles with a constant feedback loop. (Lwakatare et al., 2016; Osmundsen & Bygstad, 2022; Virmani, 2015) |
| Continuous Integration | Practice that automates and combines all source code changes by several authors into a single software project (Gall & Pigni, 2021; Stray et al., 2019). |
| Continuous Deployment | Practice that automates the release process through frequent and automated deployments (Gall & Pigni, 2021). |
| Continuous Delivery | Extension of continuous integration that aims to shorten release cycles by automating software testing and approval (Chen, 2015; Ghantous & Gill, 2017). |
| DevOps | Instantiation of CD: fast delivery cycles, short feedback loops and automation. Extends agile principles to the entire software process and adds cooperation between operations, development, and quality assessment. (Ebert, 2018; Krey et al., 2022; Lwakatare et al., 2016; Olszewska & Waldén, 2015; Osmundsen & Bygstad, 2022, Ozkaya, 2019; Stray et al., 2019) |
| Requirements Risk Management (RRM) | Focuses on the active identification and management of solution-related uncertainties. Helps anticipate and solve problems that may affect requirements during the implementation process. It is a process to handle risks associated with the requirements of a project or system. (Venkatesh et al., 2018; Wallace et al., 2004) |
| Requirements Engineering (RE) | Defines, documents, and maintains the needs and expectations of a software system. (Ramesh et al., 2010) |

*Table A1: Key concepts and definitions*

## Appendix 2. Systematic literature review (SLR)

Our SLR followed Kitchenham et al.'s (2010) guidelines (Figure 1). In Step 1, we defined the need for this study. Several IS articles have identified the need for further research on how CD and DevOps can be used in ISD. We wanted to investigate how the current IS literature covers the key elements of CD (Step 2) and sought answers to our research question (Step 3): "How should requirement risks be managed in CD, and what are the organisational needs for accomplishing this?" In Step 4, we searched Google Scholar. A more detailed description follows.

### A2.1 Finding DevOps and CD in the IS literature

A Google Scholar search (Step 4) was performed in August 2018 to obtain an overview of how the AISBASKET8 covers CD, DevOps, continuous analysis, continuous implementation, continuous integration and development and secure operations (DevSecOps). Because a simple search for "DevOps" returned only 15 articles, additional criteria were added to the search (Step 5), as follows: "continuous analysis" OR "continuous development" OR "continuous implementation" OR "continuous integration" OR "DevOps" OR "DevSecOps".

This extended search, without a defined time limit, generated 171 articles, 16 of which were selected for further processing after all the articles were read. Ten other journal and conference results were found as a result of the original search or a forward and backward reference search. Four of these articles were selected for further processing (Step 6).

The selection criteria for further processing were that the article included the search terms in the text and that the article focused on software development using agile DevOps or CD. If the search terms were mentioned only in an article's reference list, it was excluded (Step 7).

Another Google Scholar search with the same criteria was conducted on the proceedings of the following conferences: the American Conference for IS (AMCIS), the European Conference for IS (ECIS), the Hawaiian International Conference for System Sciences (HICSS), the International Conference for IS (ICIS) and the Pacific-Asian Conference IS (PACIS). This search generated 113 CD articles, of which 16 were selected for further examination.

In September 2019, an updated search (u2) on the AISBASKET8 was performed to include articles published after August 2018. This search yielded eight articles; two were chosen for further handling and coding with ATLAS.ti.

In May 2022, an updated search (u3) on the AISBASKET8 was performed to find the newest articles. Of 22 found, 9 were chosen for further handling.

From the IS literature searches, we selected 43 of 314 articles for further processing. Ten other suitable journal articles and conference proceedings were defined during the search process, and four were selected for further processing.

The Google Scholar searches for IS journals and conference proceedings returned 314 (171 + 8 + 22 + 113) articles, of which 43 (16 + 2 + 9 + 16) were selected for further processing. After adding other journals and conference proceedings, the searches returned 324 (314 + 10) items, of which 47 (43 + 4) were processed further.

### A2.2 Finding RRM in the IS literature

The next step defined how widely the selected IS journals covered RRM. The Google Scholar search on RRM without any defined time limit used the following search string: "requirement" + "risk management" -finance* -economic* -biotech* -medical* -military. The search was limited to articles from the AISBASKET8. This search (s1) generated 114 articles, of which 14 were selected after the articles were read. The selection criteria for the articles were that they covered requirements and risk management in ISD; articles that did not cover the desired concepts or mentioned search terms only in the reference list were left out.

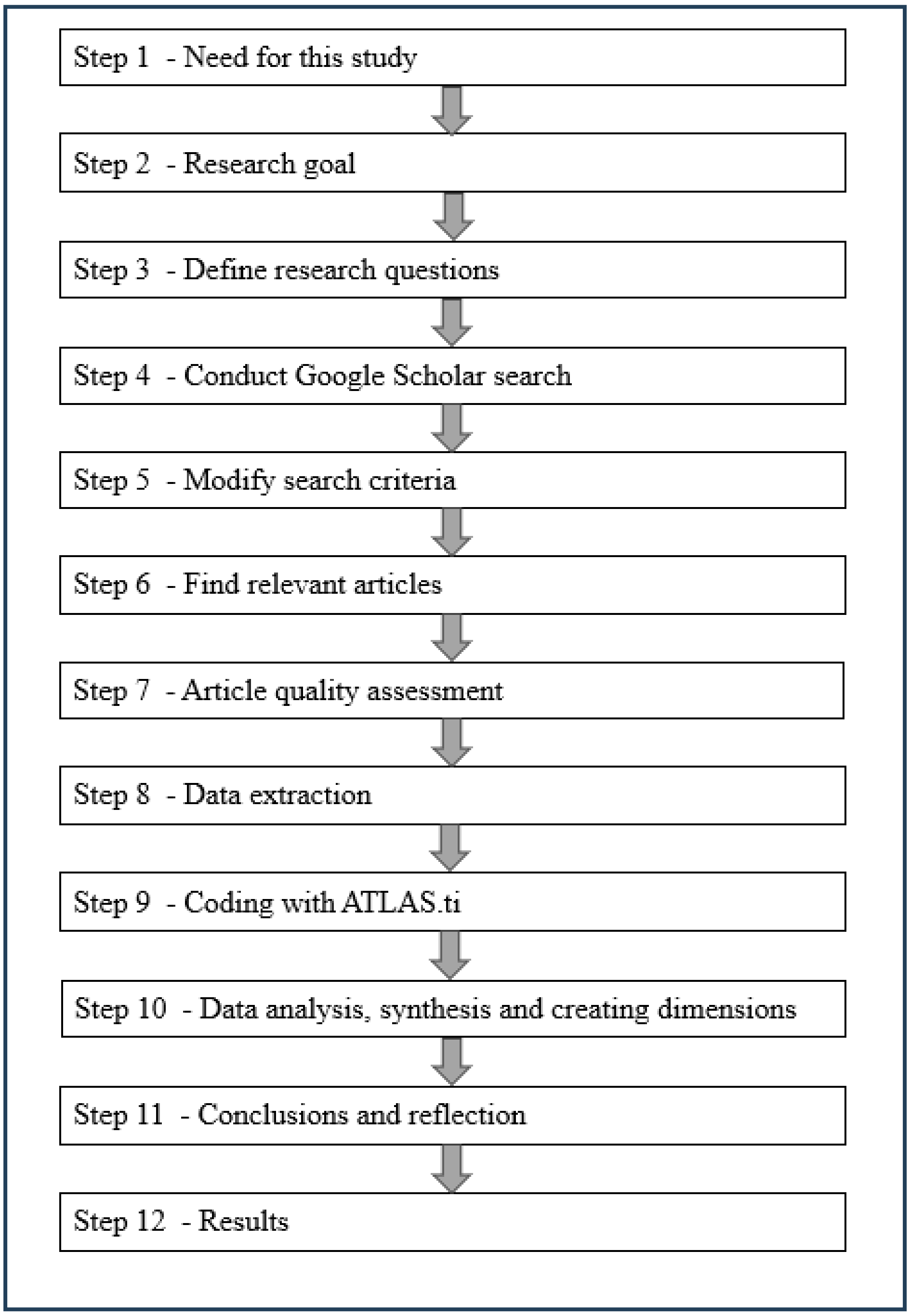
Step 1 - Need for this study
Step 2 - Research goal
Step 3 - Define research questions
Step 4 - Conduct Google Scholar search
Step 5 - Modify search criteria
Step 6 - Find relevant articles
Step 7 - Article quality assessment
Step 8 - Data extraction
Step 9 - Coding with ATLAS.ti
Step 10 - Data analysis, synthesis and creating dimensions
Step 11 - Conclusions and reflection
Step 12 - Results

***Figure A1**. Systematic literature review steps based on Kitchenham et al. (2010).*

Another Google Scholar search with the same criteria was conducted to cover knowledge of this subject in IS conferences. Articles were selected from the AMCIS, ECIS, HICSS, ICIS and PACIS proceedings. Of the 135 RRM articles generated, 10 were selected for further examination. The first two IS RRM literature searches defined 24 (of 249) articles for further processing.

In September 2019, an updated search (u2) on the AISBASKET8 was performed to include articles published after August 2018. An updated search for RRM yielded six new articles, none of which were selected for further processing.

In May 2022, an updated search (u3) on the AISBASKET8 was performed to find the newest articles. Two of the 11 recent ones were chosen for further handling.

From the search results and using a forward and backward reference search, 22 interesting and suitable articles and conference proceedings were found from publications other than the specified AISBASKET8 or IS conference proceedings, of which 10 were selected for further processing.

Based on the RRM search results from IS journals, conference proceedings and other interesting journals and conference proceedings, 36 (14 + 0 + 2 + 10 +10) articles were selected out of the 288 (114 + 135 + 6 + 11 + 22) articles. After analysing the 36 documents, 25 were selected for further processing.

### A2.3 Creating the framework dimensions

The qualitative research software ATLAS.ti was used to code the articles, group the codes and define the key features of CD. Certain concepts and terms emerged repeatedly from the literature. First, key elements (Step 8) were extracted from the articles by tagging the paper's paragraphs, and then the tagged sections were coded with ALTAS.ti (Step 9). Code groups were formed from the codes, the metadata-level concepts were defined, and the framework's three dimensions were combined (Step 10). Dimensions of the framework were determined: (1a) culture – project-related tasks, such as roles, resources, instructions and guidance, management of work, teamwork, cooperation and knowledge sharing; (1b) culture – organisation-related issues, such as organisational structure in the development process; (2) different methods and tools; and (3) pace, seamlessness and continuous work – all typical features of CD. These dimensions formed the framework used as a research lens to analyse the RRM articles (Step 10).

### A2.4 Analysing IS RRM articles with a three-dimensional research lens

The selected IS RRM articles (25 of 36) were analysed using the three-dimensional framework described above. The aim was to discover how key features of agile development, DevOps and CD were defined in IS RRM articles. Of 25 RRM articles, 16 (64%) focused on methods and tools, 12 (48%) on culture/project structure, 7 (28%) on culture/organisational structure and 4 (16%) on development speed or seamlessness. Most articles did not comment on the development approaches, other than some notes about traditional methods; 28% mentioned agile development and only 4% mentioned CD. Thus, the literature review revealed that, despite the extensive IS literature, there was still a lack of focus on CD.

### A2.5 Finding RRM, DevOps and CD in the RE literature

The same search was performed in journals on RE. The nine journals that published RE articles (Table A3) were searched without a time limit for articles that covered the desired research subject and two selected terms – RRM and CD. The search was conducted in three phases. The first phase (s1) was searching for articles with the search criterion of DevOps or CD. To reduce the number of results, the terms "requirements" and "risk" were added in the second search (s2). The criterion "software engineering" was added in the third search (s3). After three search cycles, the third search generated 159 (DevOps 86 + CD 73) articles, of which 8 (7 + 1) were selected for further processing. Those that covered DevOps, CD or agile development, where the focus was not only on the agility of publishing or deployment of software, were selected.

Many articles written from the software development perspective dealt only with the agile deployment of software versions and were thus beyond this study's scope.

## A2.6 Analysing the RE RRM articles

The selected RE articles were coded with ATLAS.ti, with very similar results to the RE articles emerging from similar concepts, thus reinforcing the concepts defined in the IS articles.

These articles were also analysed using the developed framework as a research lens. Five (63%) articles focused on methods and tools, 3 (38%) on culture/project, 4 (50%) on culture/organisation and 5 (63%) on development speed or seamlessness. Most articles commented on agile, DevOps or CD. These findings revealed that, in the RE literature, agile development, especially DevOps and CD, was recognised and studied much more than in the IS literature.

## A2.7 Comparing the analyses and refining the framework

In the SLR, we analysed articles from the IS and RE literature. Table A1 shows the Google Scholar search steps, the search results and the selected articles on requirements risk management (RRM) and continuous development (CD) from the IS literature. Table A3 presents the information from the RE literature. Table A2 lists the Google Scholar search results and the selected articles from IS conference proceedings.

The IS and RE articles showed an almost equal focus on the methods and tools dimension: 64% vs. 63%, respectively. The difference was slightly higher for the culture/project dimension: 48% vs. 38%, respectively. The most significant differences were in the culture/organisation and development speed or seamlessness dimensions. The IS articles focused much less on these dimensions than the RE articles. Only 28% of the IS articles concentrated on the elements of the culture/organisation dimension, compared to 50% of the RE articles. The difference was even more significant in the development speed or seamlessness dimension. Only 16% of the IS articles presented this dimension's elements, while in the RE articles, the coverage was 63%. There was also a clear difference in mentioning the development style. On the IS side, it was usually not mentioned for anything other than traditional methods, and there was still little focus on CD. In contrast, in RE, most articles mentioned agile development, DevOps or CD. This comparative analysis showed that agile development, especially DevOps and CD, was much more acknowledged in the RE literature than in the IS literature.

| **Journal** | **RRM** | | **CD** | |
|---|---|---|---|---|
| | **Search s1/u2/u3** | **Selected articles s1/u2/u3** | **Search s1/u2/u3** | **Selected articles s1/u2/u3** |
| European Journal of Information Systems | 10/0/0 | 1/0/0 | 16/1/7 | 3/1/6 |
| Information Systems Journal | 11/0/2 | 3/0/1 | 16/1/5 | 1/0/0 |
| Information Systems Research | 25/0/1 | 1/0/0 | 26/0/3 | 3/0/0 |
| Journal of Information Technology | 38/1/2 | 4/0/0 | 75/1/3 | 1/0/1 |
| Journal of Management Information Systems | 7/0/0 | 1/0/0 | 13/2/2 | 2/0/1 |

| | | | | |
|---|---|---|---|---|
| Journal of Strategic Information Systems | 1/1/1 | 0/0/0 | 3/1/0 | 0/0/0 |
| Journal of the Association for Information Systems | 6/2/2 | 3/0/0 | 7/1/0 | 2/1/0 |
| MIS Quarterly | 16/2/3 | 1/0/1 | 15/1/4 | 4/0/1 |
| Total AISBASKET8 | 114/6/11 | 14/0/2 | 171/8/22 | 16/2/9 |
| IS conference proceedings (Table A2) | 135 | 10 | 113 | 16 |
| Other journals/conference proceedings | 22 | 10 | 10 | 4 |
| Total (including update) | 288 | 36 | 324 | 47 |

*Table A2. Google Scholar search results and selected articles about RRM and CD*

| **Conference proceedings** | **RRM** | | **CD** | |
|---|---|---|---|---|
| | **Search results** | **Selected articles** | **Search results** | **Selected articles** |
| AMICS | 11 | 3 | 29 | 4 |
| ECIS | 10 | 2 | 17 | 2 |
| HICSS | 24 | 3 | 29 | 5 |
| ICIS | 33 | 0 | 28 | 1 |
| PACIS | 57 | 2 | 10 | 4 |
| Total | 135 | 10 | 113 | 16 |

*Table A3. Google Scholar search results for RRM and CD and selected articles from IS conference proceedings*

| **Journal** | **DevOps** | | | | **CD** | | | |
|---|---|---|---|---|---|---|---|---|
| | **Search results, s1/s2/s3** | | | **Selected articles** | **Search results, s1/s2/s3** | | | **Selected articles** |
| Empirical Software Engineering | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| IEEE Software | 98 | 39 | 39 | 6 | 13 | 10 | 7 | 1 |
| IEEE Transactions on Software Engineering | 12 | 7 | 6 | 0 | 7 | 5 | 4 | 0 |
| Information and Software Technology | 23 | 12 | 12 | 1 | 24 | 15 | 14 | 0 |
| Information Systems (IS) | 95 | 47 | 25 | 0 | 463 | 196 | 47 | 0 |
| Requirements Engineering Journal | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Software and Systems Modelling | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| Software Practice and Expertise | 13 | 5 | 2 | 0 | 4 | 2 | 1 | 0 |
| Software Quality Journal | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| Total | 243/112/86 | | | 7 | 511/228/73 | | | 1 |

*Table A4. Google Scholar search results and selected RE articles*

## Appendix 3. Coding process

As an example of the coding process, we present the definition of the first codes in the article *"Key Affordances of Platform-as-a-Service: Self-Organisation and Continuous Feedback"* by Krancher, O., Luther, P. and Jost, M., published in 2018 in the *Journal on Management Information Systems*. We present the first highlighted texts we coded, from which the base codes were defined and then selected and added to the code groups to specify the dimensions.

Quotation 1: "*… rapidly deliver innovative software, many software development teams attempt to follow movements such as agile [29] and lean [62] software development*[2]*, continuous integration and delivery [41], and DevOps*[1] *[2]. Common to these movements is the aim to increase agility*[4] *(i.e., the ability to rapidly create, react to, and learn from change [19]) by adopting practices based on self-organizing*[5] *and frequent feedback*[3] *[28, 73]*".

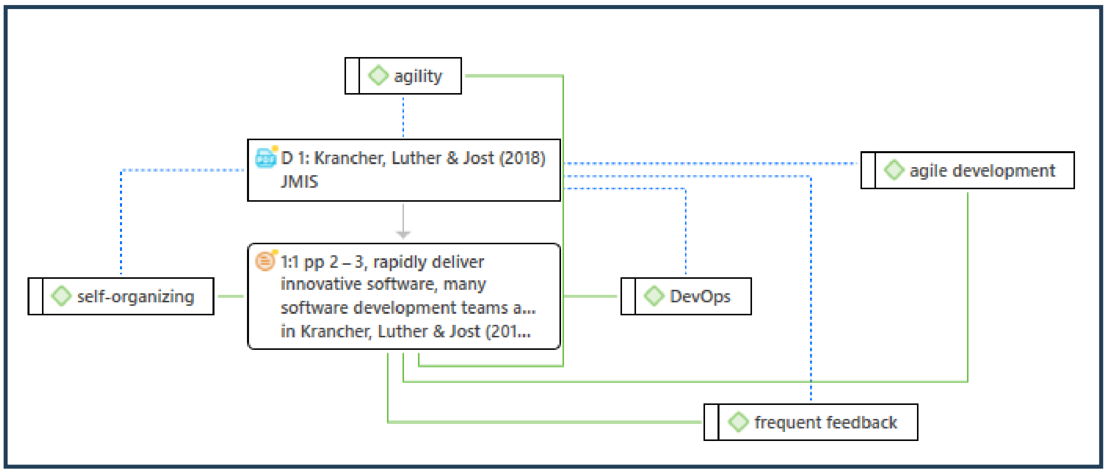
agility
D 1: Krancher, Luther & Jost (2018) JMIS
agile development
1:1 pp 2 – 3, rapidly deliver innovative software, many software development teams a... in Krancher, Luther & Jost (201...
self-organizing
DevOps
frequent feedback

*Figure A2: Codes: DevOps*[1]*, agile development*[2]*, frequent feedback*[3]*, agility*[4]*, self-organizing*[5]

Quotation 2: "*Practices based on self-organizing*[6] *are a hallmark of the agile software development movement, which values "individuals and interactions over [externally imposed] processes" [29] and which advocates uniting business users and developers*[7,9] *in a self-organizing team [29]. These teams make decisions about requirements*[10]*, solution designs, and the distribution of work [66]. Self-organizing is also a key idea behind DevOps*[8]*, which advocates joint teams of developers and system administrators*[7] *with no rigid separation of roles*[11] *between the two [42]*".

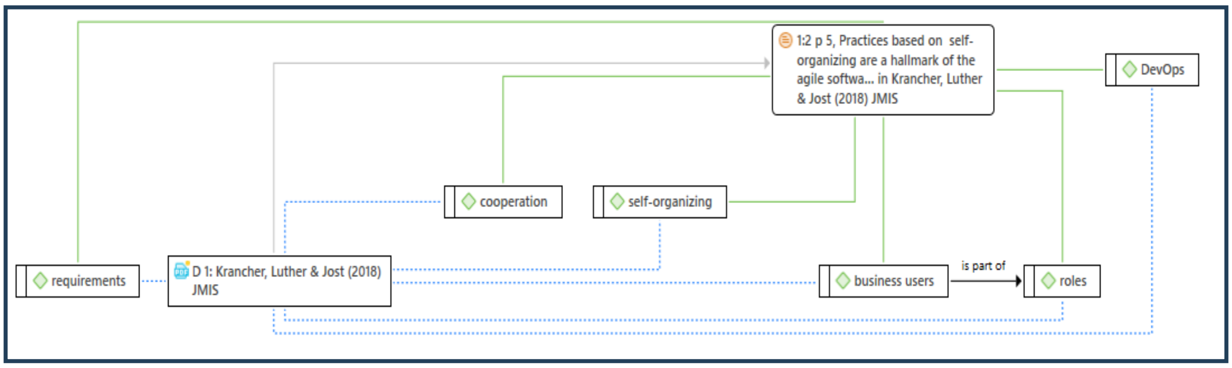
1:2 p 5, Practices based on self-organizing are a hallmark of the agile softwa... in Krancher, Luther & Jost (2018) JMIS
DevOps
cooperation
self-organizing
requirements
D 1: Krancher, Luther & Jost (2018) JMIS
business users
is part of
roles

*Figure A3: Codes: self-organizing*[6]*, cooperation*[7]*, DevOps*[8]*, business users*[9]*, requirements*[10]*, roles*[11]

Quotation 3: *"… as realized by the DevOps12 movement, software development teams often lack control over infrastructure13 and knowledge14 to manage infrastructure13".*

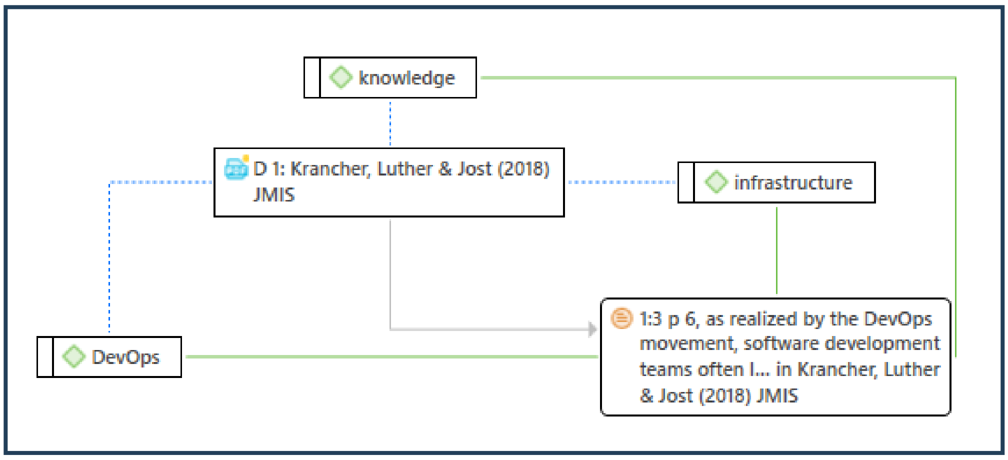
knowledge
D 1: Krancher, Luther & Jost (2018) JMIS
infrastructure
1:3 p 6, as realized by the DevOps movement, software development teams often l… in Krancher, Luther & Jost (2018) JMIS
DevOps

*Figure A4: Codes: DevOps[12], infrastructure[13], knowledge[14]*

Quotation 4: *"While the <u>DevOps</u>[17] movement suggests removing this <u>barrier by including system administrators into development teams and by eliminating the rigid separation</u>[15] <u>of roles</u>[18], empirical evidence shows that developers and operations often take their traditional division of labor for granted [63]. In such teams, the transition to self-organizing practices is a relatively slow process of <u>cultural change</u>[16]".*

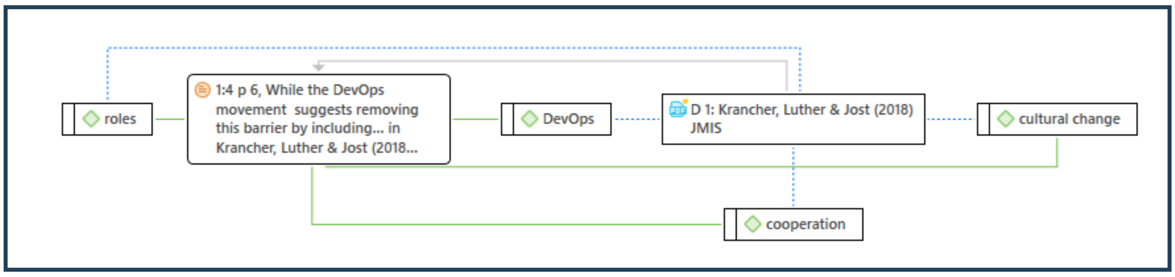
roles
1:4 p 6, While the DevOps movement suggests removing this barrier by including… in Krancher, Luther & Jost (2018…
DevOps
D 1: Krancher, Luther & Jost (2018) JMIS
cultural change
cooperation

*Figure A5: Codes: cooperation[15], cultural change[16], DevOps[17], roles[18]*

Quotation 5: *"Lean methods acknowledge that such time-boxed rhythms may still delay <u>feedback</u>[21]. They recommend further "increas[ing] the frequency of the <u>feedback loops</u>[22]" [62, p. 38] because "the shorter these <u>cycles are, the more can be learned</u>[23]" [62, p. 14]. Principle is put into practice by the <u>continuous integration</u>[20] and <u>continuous delivery</u>[19] (CI/CD) movement".*

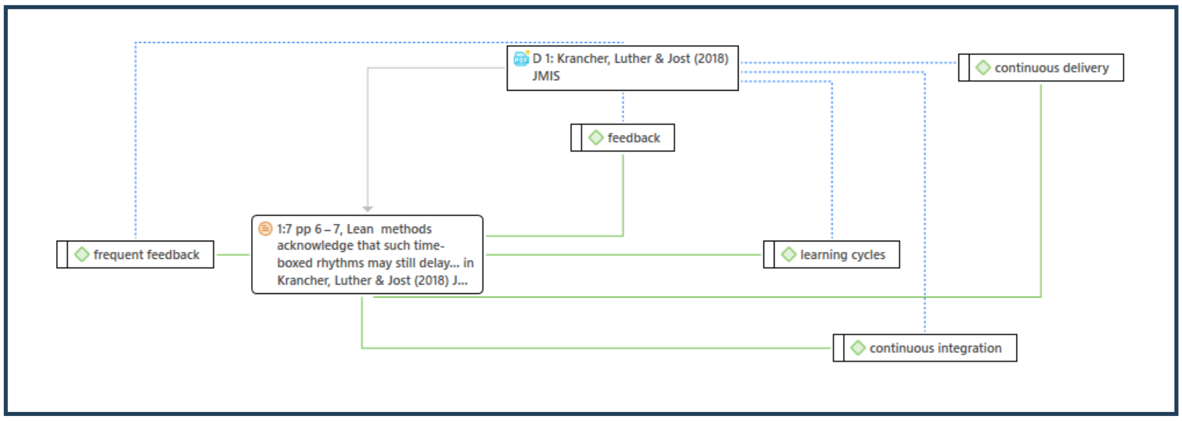
D 1: Krancher, Luther & Jost (2018) JMIS
continuous delivery
feedback
frequent feedback
1:7 pp 6 – 7, Lean methods acknowledge that such time-boxed rhythms may still delay… in Krancher, Luther & Jost (2018) J…
learning cycles
continuous integration

*Figure A6: Codes: continuous delivery[19], continuous integration[20], feedback[21], feedback loops[22], learning cycles[23]*

Quotation 6: *"… agile[26] methods advocate frequent feedback[24] through time-boxed iterations, often of a duration of a few weeks [73], which are seen as "learning cycles[25]".*

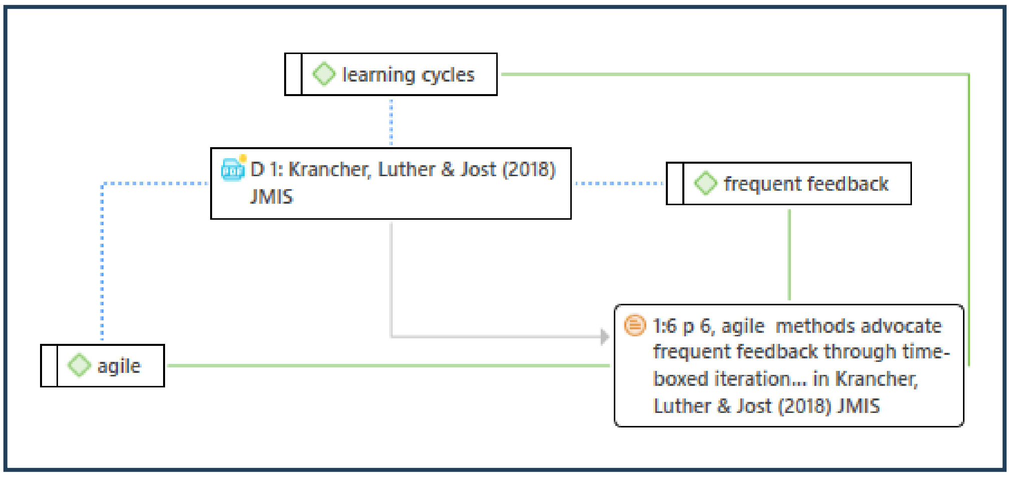
learning cycles
D 1: Krancher, Luther & Jost (2018) JMIS
frequent feedback
agile
1:6 p 6, agile methods advocate frequent feedback through time-boxed iteration... in Krancher, Luther & Jost (2018) JMIS

*Figure A7: Codes: frequent feedback[24], learning cycles[25], agile[26]*

Selected codes were grouped into code groups, which were used to define dimensions. Table A5 presents the codes, code groups and dimensions.

| Code | Code Group | Dimension |
|---|---|---|
| Business users | Project characteristics | 1a) Culture – Project Culture |
| Business value | Project characteristics | 1a) Culture – Project Culture |
| Business needs | Project characteristics | 1a) Culture – Project Culture |
| Cooperation | Project characteristics | 1a) Culture – Project Culture |
| Collaboration | Project characteristics | 1a) Culture – Project Culture |
| Communication | Project characteristics | 1a) Culture – Project Culture |
| Coordination | Project characteristics | 1a) Culture – Project Culture |
| Customer engagement | Project characteristics | 1a) Culture – Project Culture |
| Customer representative | Project characteristics | 1a) Culture – Project Culture |
| Feedback | Project characteristics | 1a) Culture – Project Culture |
| Frequent feedback | Project characteristics | 1a) Culture – Project Culture |
| Interaction | Project characteristics | 1a) Culture – Project Culture |
| Knowledge | Project characteristics | 1a) Culture – Project Culture |
| Knowledge sharing | Project characteristics | 1a) Culture – Project Culture |
| Learning cycles | Project characteristics | 1a) Culture – Project Culture |
| Obstacle | Project characteristics | 1a) Culture – Project Culture |
| Performance | Project characteristics | 1a) Culture – Project Culture |
| Project management | Project characteristics | 1a) Culture – Project Culture |

| | | |
|---|---|---|
| Requirements | Project characteristics | 1a) Culture – Project Culture |
| Resources | Project characteristics | 1a) Culture – Project Culture |
| Risk level | Project characteristics | 1a) Culture – Project Culture |
| Risk management | Project characteristics | 1a) Culture – Project Culture |
| Roles | Project characteristics | 1a) Culture – Project Culture |
| Threat | Project characteristics | 1a) Culture – Project Culture |
| Users and user involvement | Project characteristics | 1a) Culture – Project Culture |
| Business environment | Organisation structure | 1b) Culture – Organisational Culture |
| Infrastructure | Organisation structure | 1b) Culture – Organisational Culture |
| Organisation | Organisation structure | 1b) Culture – Organisational Culture |
| Work environment | Organisation structure | 1b) Culture – Organisational Culture |
| Business agility | Organisation structure | 1b) Culture – Organisational Culture |
| Cultural change | Organisation structure | 1b) Culture – Organisational Culture |
| Organisational culture | Organisation structure | 1b) Culture – Organisational Culture |
| Organisational change | Organisation structure | 1b) Culture – Organisational Culture |
| Lack of knowledge of DevOps | Continuous development | 2) Methods and Tools |
| Lack of guidance | Continuous development | 2) Methods and Tools |
| Lack of research | Continuous development | 2) Methods and Tools |
| Agile development | Development style | 2) Methods and Tools |
| Agile principles and methodology | Development style | 2) Methods and Tools |
| Agile to DevOps | Development style | 2) Methods and Tools |
| Continuous delivery | Development style | 2) Methods and Tools |
| Continuous deployment | Development style | 2) Methods and Tools |
| Continuous development | Development style | 2) Methods and Tools |
| Continuous integration | Development style | 2) Methods and Tools |
| Definition of DevOps | Development style | 2) Methods and Tools |
| Development cycle | Development style | 2) Methods and Tools |
| DevOps | Development style | 2) Methods and Tools |
| Fast agile development | Development style | 2) Methods and Tools |
| Method engineering | Development style | 2) Methods and Tools |
| Software development | Development style | 2) Methods and Tools |
| Traditional developing | Development style | 2) Methods and Tools |

| | | |
|---|---|---|
| Behaviour-driven monitoring | Features and components | 2) Methods and Tools |
| Shared goal | Features and components | 2) Methods and Tools |
| Tools | Features and components | 2) Methods and Tools |
| Flexibility | Pace and seamlessness | 3) Pace and seamlessness |
| Quickness | Pace and seamlessness | 3) Pace and seamlessness |
| Speed | Pace and seamlessness | 3) Pace and seamlessness |

***Table A5.*** *Codes, code groups and dimensions*

## Appendix 4. Analysed and referenced articles

| **European Journal of Information Systems** |
|---|
| 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. https://doi.org/10.1057/ejis.2009.26 |
| Gall, M., & Pigni, F. (2021). Taking DevOps mainstream: A critical review and conceptual framework. *European Journal of Information Systems*, *31*(5), 548–567. https://doi.org/10.1080/0960085X.2021.1997100 |
| Hemon-Hildgen, A., Rowe, F., & Monnier-Senicourt, L. (2020). Orchestrating automation and sharing in DevOps teams: A revelatory case of job satisfaction factors, risk and work conditions. *European Journal of Information Systems*, *29*(5), 474–499. https://doi.org/10.1080/0960085X.2020.1782276 |
| 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. https://doi.org/10.1057/ejis.2009.23 |
| Maruping, L. M., & Matook, S. (2020a). The evolution of software development orchestration: Current state and an agenda for future research. *European Journal of Information Systems*, *29*(5), 443–457. https://doi.org/10.1080/0960085X.2020.1831834 |
| Mathiassen, L., & Pries-Heje, J. (2006). Business agility and diffusion of information technology. *European Journal of Information Systems*, *15*(2), 116–119. https://doi.org/10.1057/palgrave.ejis.3000610 |
| Rowe, F. (2014). What literature review is not: Diversity, boundaries and recommendations. *European Journal of Information Systems*, *23*(3), 241–255. https://doi.org/10.1057/ejis.2014.7 |
| Shimada, T., Ang Soo-Keng, J., & Ee, D. (2019). Exploring the impact of IS function maturity and IS planning process on IS planning success: An ACE analysis. *European Journal of Information Systems*, *28*(4), 457–472. https://doi.org/10.1080/0960085X.2018.1557373 |
| Tuunanen, T., & Kuo, I. T. (2015). The effect of culture on requirements: A value-based view of prioritization. *European Journal of Information Systems*, *24*(3), 295–313. https://doi.org/10.1057/ejis.2014.29 |
| 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*, *29*(5), 458–473. https://doi.org/10.1080/0960085X.2020.1782277 |
| **Journal of the Association for Information Systems** |
| Bragge, J., & Merisalo-Rantanen, H. (2009). Engineering e-collaboration processes to obtain innovative end-user feedback on advanced web-based information systems. *Journal of the Association for Information Systems*, *10*(3), 196–220. https://doi.org/10.17705/1jais.00188 |

| |
|---|
| Davern, M., Shaft, T., & Te'eni, D. (2012). More enduring questions in cognitive IS research: A reply. *Journal of the Association for Information Systems*, *13*(12), 1012–1016. https://doi.org/10.17705/1jais.00317 |
| Jiang, J. J., Klein, G., & Chen, H. G. (2006). The effects of user partnering and user non-support on project performance. *Journal of the Association for Information Systems*, *7*(2), 68–90. https://doi.org/10.17705/1jais.00082 |
| Li, E. Y., Jiang, J. J., & Klein, G. (2003). The impact of organizational coordination and climate on marketing executives' satisfaction with information systems services. *Journal of the Association for Information Systems*, *4*(1), 99–117. https://doi.org/10.17705/1jais.00031 |
| Patnayakuni, R, Ruppel, C. P., & Rai, A. (2006). Managing the complementarity of knowledge integration and process formalization for systems development performance. *Journal of the Association for Information Systems*, *7*(8), 545–567. https://www.proquest.com/scholarly-journals/managing-complementarity-knowledge-integration/docview/198858662/se-2 |
| Salmela, H., Baiyere, A., Tapanainen, T., & Galliers, R. D. (2022). Digital agility: Conceptualizing agility for the digital era. *Journal of the Association for Information Systems*, *23*(5), 1080-1101. https://doi.org/10.17705/1jais.00767 |
| Xiao, X., Lindberg, A., Hansen, S., & Lyytinen, K. (2018). 'Computing' requirements for open source software: A distributed cognitive approach. *Journal of the Association for Information Systems*, *19*(12), 1217–1252. https://doi.org/10.17705/1jais.00525 |
| **Information Systems Journal** |
| Baham, C., & Hirschheim, R. (2022). Issues, challenges, and a proposed theoretical core of agile software development research. *Information Systems Journal*, *32*(1), 103-129. https://doi.org/10.1111/isj.12336 |
| Davison, R. M. (2017). Editorial: The limitations of limitations. *Information Systems Journal*, *27*(6), 695–697. https://doi.org/10.1111/isj.12167 |
| Ghobadi, S., & Mathiassen, L. (2016). Perceived barriers to effective knowledge sharing in agile software teams. *Information Systems Journal*, *26*(2), 95–125. https://doi.org/10.1111/isj.12053 |
| Ghobadi, S., & Mathiassen, L. (2017). Risks to effective knowledge sharing in agile software teams: A model for assessing and mitigating risks. *Information Systems Journal*, *27*(6), 699–731. https://doi.org/10.1111/isj.12117 |
| Kautz, K., Madsen, S., and Nørbjerg, J. (2007). Persistent problems and practices in information systems development. *Information Systems Journal*, *17*(3), 217–239. https://doi.org/10.1111/j.1365-2575.2007.00222.x |
| Keil, M., Tiwana, A., & Bush, A. (2002). Reconciling user and project manager perceptions of IT project risk: A Delphi study. *Information Systems Journal*, *12*(2), 103–119. https://doi.org/10.1046/j.1365-2575.2002.00121.x |
| Lee, J. S., Keil, M., & Wong, K. F. E. (2021). When a growth mindset can backfire and cause escalation of commitment to a troubled information technology project. *Information Systems Journal*, *31*(1), 7–32. https://doi.org/10.1111/isj.12287 |
| Qureshi, I., Fang, Y., Haggerty, N., Compeau, D. R., & Zhang, X. (2018). IT-mediated social interactions and knowledge sharing: Role of competence-based trust and background heterogeneity. *Information Systems Journal*, *28*(5), 929–955. https://doi.org/10.1111/isj.12181 |
| 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 |
| **MIS Quarterly** |
| Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS Quarterly*, *28*(1), 75–105. https://doi.org/10.2307/25148625 |

| |
|---|
| Maruping, L. M., & Matook, S. (2020b). The multiplex nature of the customer representative role in agile information systems development. *MIS Quarterly*, *44*(3), 1411–1437. https://doi.org/10.25300/MISQ/2020/12284 |
| Niederman, F., Brancheau, J. C., & Wetherbe, J. C. (1991). Information systems management issues for the 1990s. *MIS Quarterly*, *15*(4), 475–500. https://doi.org/10.2307/249452 |
| 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. https://doi.org/10.25300/MISQ/2013/37.1.02 |
| **MISQ Executive** |
| Dremel, C., Wulf, J., Herterich, M. M., Waizmann, J. C., & Brenner, W. (2017) How AUDI AG established big data analytics in its digital transformation. *MIS Quarterly Executive*, *16*(2), Article 3. https://aisel.aisnet.org/misqe/vol16/iss2/3 |
| Matook, S., & Maruping, L. M. (2014). A competency model for customer representatives in agile software development projects. *MIS Quarterly Executive*, *13*(2), Article 3. https://aisel.aisnet.org/misqe/vol13/iss2/3 |
| Sebastian, I. M., Ross, J. W., Beath, C., Mocker, M., Moloney, K. G., & Fonstad, N. O. (2017). How big old companies navigate digital transformation. *MIS Quarterly Executive*, *16*(3), Article 6. https://aisel.aisnet.org/misqe/vol16/iss3/6 |
| **Journal of Management Information Systems** |
| Gemino, A., Reich, B. H., & Sauer, C. (2007). A temporal model of information technology project performance. *Journal of Management Information Systems*, *24*(3), 9–44. https://doi.org/10.2753/MIS0742-1222240301 |
| Hickey, A. M., & Davis, A. M. (2004). A unified model of requirements elicitation. *Journal of Management Information Systems*, *20*(4), 65–84. https://doi.org/10.1080/07421222.2004.11045786 |
| 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 |
| Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of Management Information Systems*, *24*(3), 45–77. https://doi-org.ezproxy.jyu.fi/10.2753/MIS0742-1222240302 |
| Schmidt, R., Lyytinen, K., Keil, M., & Cule, P. (2001). Identifying software project risks: An international Delphi study. *Journal of Management Information Systems*, *17*(4), 5–36. https://doi.org/10.1080/07421222.2001.11045662 |
| **Journal of Information Technology** |
| Heemstra, F. J., & Kusters, R. J. (1996). Dealing with risk: A practical approach. *Journal of Information Technology*, *11*(4), 333–346. https://doi.org/10.1177/026839629601100407 |
| Taylor, H., Artman, E., & Woelfer, J. P. (2012). Information technology project risk management: Bridging the gap between research and practice. *Journal of Information Technology*, *27*(1), 17–34. https://doi.org/10.1057/jit.2011.29 |
| Osmundsen, K., & Bygstad, B. (2022). Making sense of continuous development of digital infrastructures. *Journal of Information Technology*, 37(2), 144–164. https://doi.org/10.1177/02683962211046621 |
| **Information Systems Research** |
| Maruping, L. M., 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 |

| |
|---|
| Venkatesh, V., Rai, A., & Maruping, L. M. (2018). Information systems projects and individual developer outcomes: Role of project managers and process control. *Information Systems Research*, *29*(1), 127–148. https://doi.org/10.1287/isre.2017.0723 |
| **Journal of Association of Information Systems** |
| Mathiassen, L., Saarinen, T., Tuunanen, T., & Rossi, M. (2007). A contingency model for requirements development. *Journal of Association of Information Systems*, *8*(11), 569–597. https://doi.org/10.17705/1jais.00143 |
| **IEEE Access** |
| Khan, M. S., Khan, A. W., Khan, F., Khan, M. A., & Whangbo, T. K. (2022). Critical challenges to adopt DevOps culture in software organizations: A systematic review. *IEEE Access*, *10*, 14339–14349. https://doi.org/10.1109/ACCESS.2022.3145970 |
| **IEEE Software** |
| Callanan, M., & Spillane, A. (2016). DevOps: Making it easy to do the right thing. *IEEE Software*, *33*(3), 53–59. https://doi.org/10.1109/MS.2016.66 |
| Cao, L., & Ramesh, B. (2008). Agile requirements engineering practices: An empirical study. *IEEE Software*, *25*(1), 60–67. https://doi.org/10.1109/MS.2008.1 |
| Chen, L. (2015). Continuous delivery: Huge benefits, but challenges too. *IEEE Software*, *32*(2), 50–54. https://doi.org/10.1109/MS.2015.27 |
| Dingsøyr, T., Falessi, D., & Power, K. (2019). Agile development at scale: The next frontier. *IEEE Software*, *36*(2), 30–38. https://doi.org/10.1109/MS.2018.2884884 |
| Ebert, C. (2018). 50 years of software engineering: Progress and perils. *IEEE Software*, *35*(5), 94–101. https://doi.org/10.1109/MS.2018.3571228 |
| Elbanna, A., & Sarker, S. (2015). The risks of agile software development: Learning from adopters. *IEEE Software*, *33*(5), 72–79. https://doi.org/10.1109/MS.2015.150 |
| Gatrell, M. (2016). The value of a single solution for end-to-end ALM tool support. *IEEE Software*, *33*(5), 103–105. https://doi.org/10.1109/MS.2016.109 |
| Ozkaya, I. (2019). Are DevOps and automation our next silver bullet? *IEEE Software*, *36*(4), 3–5. https://doi.org/10.1109/MS.2019.2910943 |
| **Information and Software Technology** |
| Kitchenham, B., Pretorius, R., Budgen, D., Brereton, O. P., Turner, M., Niazi, M., & Linkman, S. (2010). Systematic literature reviews in software engineering—A tertiary study. *Information and Software Technology*, *52*(8), 792–805. https://doi.org/10.1016/j.infsof.2010.03.006 |
| Lwakatare, L. E., Kilamo, T., Karvonen, T., Sauvola, T., Heikkilä, V., Itkonen, J., Kuvaja, P., Mikkonen, T., Oivo, M., & Lassenius, C. (2019). DevOps in practice: A multiple case study of five companies. *Information and Software Technology*, *114*(1), 217–230. https://doi.org/10.1016/j.infsof.2019.06.010 |
| **Journal of Database Management** |
| Siau, K., Long, Y., and Ling, M. (2010). Toward a unified model of information systems development success. *Journal of Database Management*, *21*(1), 80-101. https://doi.org/10.4018/jdm.2010112304 |
| **Communications of the ACM** |
| Tiwana, A., & Keil, M. (2004). The one-minute risk assessment tool. *Communications of the ACM*, *47*(11), 73–77. https://doi.org/10.1145/1029496.1029497 |
| **Communications of the Association for Information Systems** |

| |
|---|
| Tuunanen, T., Vartiainen, T., Kainulainen, S., & Ebrahim, M. (2023). Development of an Agile Requirements Risk Prioritization Method: A Design Science Research Study. *Communications of the Association for Information Systems*, *52*(1), 609-637. https://doi.org/10.17705/1CAIS.05226 |
| **Decision Sciences** |
| Wallace, L., Keil, M., & Rai, A. (2004). How software project risk affects project performance: An investigation of the dimensions of risk and an exploratory model. *Decision Sciences*, *35*(2), 289–321. https://doi.org/10.1111/j.00117315.2004.02059.x |
| **Empirical Software Engineering** |
| Karlsson, L., Thelin, T., Regnell, B., Berander, P., & Wohlin, C. (2007). Pair-wise comparisons versus planning game partitioning – experiments on requirements prioritization techniques. *Empirical Software Engineering*, *12*(1), 3–33. https://doi.org/10.1007/s10664-006-7240-4 |
| **IBM Systems Journal** |
| Davis, G. B. (1982). Strategies for information requirements determination, *IBM Systems Journal*, *21*(1), 4–30. https://doi.org/10.1147/sj.211.0004 |
| **Computer** |
| Highsmith, J., & Cockburn, A. (2009). Agile software development: The business of innovation. *Computer (Long Beach, Calif.)*, *34*(9), 120–127. https://doi.org/10.1109/2.947100 |
| **Future Internet** |
| Almeida, F., Simões, J., & Lopes, S. (2022). Exploring the benefits of combining devops and agile. *Future Internet*, 14(2), 63. https://doi.org/10.3390/fi14020063 |
| **International Journal of Information Systems and Project Management** |
| Jayakody, V., & Wijayanayake, J. (2023). Critical success factors for DevOps adoption in information systems development. *International Journal of Information Systems and Project Management*, 11(3), 60-82. https://doi.org/10.12821/ijispm110304 |
| **Cogent Engineering** |
| Jha, A. V., Teri, R., Verma, S., Tarafder, S., Bhowmik, W., Kumar Mishra, S., Appasani, B., Srinivasulu, A., & Philibert, N. (2023). From theory to practice: Understanding DevOps culture and mindset. *Cogent Engineering*, *10*(1), 1-31. https://doi.org/10.1080/23311916.2023.2251758 |
| **Computing in Science & Engineering** |
| Sletholt, M., Hannay, J. E., Langtangen, H. P., & Pfahl, D. (2012). What do we know about scientific software development's Agile practices? *Computing in Science & Engineering*, *14*(2), 24–37. https://doi.org/10.1109/MCSE.2011.113 |
| **Conference Proceedings** |
| Azizi, N., & Rowlands, B. (2018, June 23-28). *The moderating effects of organisational culture on the relationship between knowledge sharing and IT risk management success* [Research-in-Progress Papers]. 26th European Conference on Information Systems (ECIS), Portsmouth, United Kingdom. https://aisel.aisnet.org/ecis2018_rip/39 |
| Babb, J. S., Nørbjerg, J., & Yates, D. J. (2017, August 6-9). *The empire strikes back: The end of agile as we know it?* [Paper presentation]. Selected papers of the IRIS, Halden, Norway. http://aisel.aisnet.org/iris2017/8 |
| Chen, H-M., Kazman, R., & Haziyev, S. (2016, January 5-8). *Agile big data analytics development: An architecture-centric approach* [Paper presentation]. 49th Hawaii International Conference on System Sciences (HICSS), Koloa, HI, USA. https://doi.org/10.1109/HICSS.2016.665 |

Cois, C. A., Yankel, J., & Connell, A. (2014, October 13-15). *Modern DevOps: Optimizing software development through effective system interactions* [Paper presentation]. 2014 IEEE International Professional Communication Conference (IPCC), Pittsburgh. PA, USA. https://doi.org/10.1109/IPCC.2014.7020388

Gantman, S. (2011, August 4-8). *Boundary objects and internal control in outsourced ISD projects: Results of a pilot study* [Paper presentation]. 17th Americas Conference on Information Systems (AMCIS), 3, Detroit, Michigan, USA. https://aisel.aisnet.org/amcis2011_submissions/200

Ghanbari, H. (2016, January 5-8). *Seeking technical debt in critical software development projects: An exploratory field study* [Paper presentation]. 49th Hawaii International Conference on System Sciences (HICSS) ), Koloa, HI, USA. https://doi.org/10.1109/HICSS.2016.668

Ghantous, G. B., & Gill, A. (2017, July 16-20). *DevOps: Concepts, practices, tools, benefits and challenges* [Paper presentation]. 21st Pacific Asia Conference on Information Systems (PACIS), Langkawi, Malaysia. https://aisel.aisnet.org/pacis2017/96

Horlach, B., Drews, P., Drechsler, A., Schirmer, I., & Böhmann, T. (2020, June 15-17). *Reconceptualising business-IT alignment for enabling organizational agility* [Paper presentation]. 28th European Conference on Information Systems (ECIS), Marrakech, Morrocco. https://aisel.aisnet.org/ecis2020_rp/95

Hüttermann, M., & Rosenkranz, C. (2019, December 15-18). *DevOps: Walking the shadowy bridge from development success to information systems success* [Paper presentation]. 40th International Conference on Information Systems (ICIS), Munich, Germany. https://aisel.aisnet.org/icis2019/is_development/is_development/10

Iyawa, G. E. (2020, August 10-14). *Personal extreme programming: Exploring developers' adoption* [Paper presentation]. 26th Americas Conference on Information Systems (AMCIS), Salt Lake City, Utah, USA. https://aisel.aisnet.org/amcis2020/it_project_mgmt/it_project_mgmt/1

Kiper, J. R. (2016, January 5-8). *Needs to know: Validating user needs for a proposed FBI Academy Knowledge Management System* [Paper presentation]. 49th Hawaii International Conference on System Sciences (HICSS), Koloa, HI, USA. https://doi.org/10.1109/HICSS.2016.538

Krey, M., Kabbout, A., Osmani, L., & Saliji, A. (2022, January 4-7). *DevOps adoption: Challenges & barriers* [Paper presentation]. 55th Hawaii International Conference on System Sciences (HICSS), Maui, HI, USA. http://hdl.handle.net/10125/80219

Lwakatare, L. E., Karvonen, T., Sauvola, T., Kuvaja, P., Olsson, H. H., Bosch, J., & Oivo, M. (2016, January 5-8). *Towards DevOps in the embedded systems domain: Why is it so hard?* [Paper presentation]. 49th Hawaii International Conference on System Sciences (HICSS), Koloa, HI, USA. https://doi.org/10.1109/HICSS.2016.671

Olszewska, M., & Waldén, M. (2015, September 1). *DevOps meets formal modelling in high-criticality complex systems* [Paper presentation]. 1st International Workshop on Quality-Aware DevOps (QUDOS 2015), Bergamo, Italy. https://doi.org/10.1145/2804371.2804373

Racheva, Z., Daneva, M., Herrmann, A., & Wieringa, R. J. (2010, May 19-21). *A conceptual model and process for client-driven agile requirements prioritization* [Paper presentation]. 2010 Fourth International Conference on Research Challenges in Information Science (RCIS 2010), Nice, France. https://doi.org/10.1109/RCIS.2010.5507388

Royce, W. W. (1987, March 30 - April 2). *Managing the development of large software systems: Concepts and techniques* [Paper presentation]. 9th International Conference of Software Engineering (ICSE '87), Monterey, California, USA. https://doi.org/10.5555/41765.41801

Sharp, J., & Babb, J. (2018, August 16-18). *Is information systems late to the party? The current state of DevOps research in the Association for Information Systems eLibrary* [Paper presentation]. 24th Americas Conference on Information Systems (AMCIS), New Orleans, LA, USA. https://aisel.aisnet.org/amcis2018/AdvancesIS/Presentations/26

| | |
|---|---|
| Stray, V., Moe, N. B., & Aasheim, A. (2019, January 8-11). *Dependency management in large-scale agile: A case study of DevOps teams* [Paper presentation]. 52nd Hawaii International Conference on System Sciences (HICSS), Maui, HI, USA. http://hdl.handle.net/10125/60137 | |
| Stuckenberg, S., & Heinzl, A. (2010, July 9-12). *The impact of the software-as-a-service concept on the underlying software and service development processes* [Paper presentation]. Pacific Asia Conference on Information Systems (PACIS), Taipei, Taiwan. https://aisel.aisnet.org/pacis2010/125 | |
| Virmani, M. (2015, May 20-22). *Understanding DevOps & bridging the gap from continuous integration to continuous delivery* [Paper presentation]. 5th International Conference on the Innovative Computing Technology (INTECH 2015), Galicia, Spain. https://doi.org/10.1109/INTECH.2015.7173368 | |
| Wang, S. Y., Chang, T. H., Hsu, J. S. C., & Lin, T. C. (2016, June 27-July 1). *A study of the influences of knowledge boundary spanning on project performance in information system development projects* [Paper presentation]. Pacific Asia Conference on Information Systems (PACIS), Chiayi, Taiwan. http://aisel.aisnet.org/pacis2016/135 | |
| **Books** | |
| Hütterman, M. (2012) | *DevOps for developers*. Apress. https://doi.org/10.1007/978-1-4302-4570-4 |

*Table A6: Analysed and referenced articles*