Association for Information Systems

AIS Electronic Library (AISeL)

14th Scandinavian Conference on Information Systems

Scandinavian Conference on Information Systems

9-22-2023

EXPLAINING SOFTWARE PROJECT RISKS IN AGILE TEAMS: AN ACTION CASE USING CAUSAL MAPPING

David Kinnberg Hein
Aalborg University, davidkh@cs.aau.dk

John Stouby Persson
Aalborg University, john@cs.aau.dk

Peter Axel Nielsen
Aalborg University, pan@cs.aau.dk

Follow this and additional works at: https://aisel.aisnet.org/scis2023

Recommended Citation

Hein, David Kinnberg; Persson, John Stouby; and Nielsen, Peter Axel, "EXPLAINING SOFTWARE PROJECT RISKS IN AGILE TEAMS: AN ACTION CASE USING CAUSAL MAPPING" (2023). *14th Scandinavian Conference on Information Systems*. 4.
https://aisel.aisnet.org/scis2023/4

This material is brought to you by the Scandinavian Conference on Information Systems at AIS Electronic Library (AISeL). It has been accepted for inclusion in 14th Scandinavian Conference on Information Systems by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.

EXPLAINING SOFTWARE PROJECT RISKS IN AGILE TEAMS: AN ACTION CASE USING CAUSAL MAPPING

Research Paper

David Kinnberg Hein, Department of Computer Science, Aalborg University, Denmark, davidkh@cs.aau.dk

John Stouby Persson, Department of Computer Science, Aalborg University, Denmark, john@cs.aau.dk

Peter Axel Nielsen, Department of Computer Science, Aalborg University, Denmark, pan@cs.aau.dk

Abstract

Agile teams must deal with many risks to their software project's resources and schedule. This challenge is exacerbated in large-scale agile development by the increasingly specialized team roles with inherently different explanations of the software project's risks. Against this backdrop, we report an action case study of how an agile team can understand and manage different explanations of their project risks. We used the causal mapping technique to understand how a team's six different roles explain their software project risks and assessed these maps' usefulness with the team. From this action case study, we present two findings. First, causal mapping is useful for revealing role-specific explanations of software project risks in agile teams. Second, agile teams can use role-specific causal maps to juxtapose their explanations of software project risks. We discuss how these findings imply a caveat for agile teams seeking to understand and manage team-generalized software project risks and ignoring idiosyncratic explanations of software project risks.

Keywords: Causal Mapping, Software Project Risks, Agile Team, Risk Assessment, Action Case Study.

1 Introduction

Software teams face an inescapable need to reduce the chance of project failure. Still, ignoring project risks remains a major recurring cause of project failure (Tavares et al., 2021). Software project risks are conditions that influence the project schedule, resources, and success (Hauck & Vieira, 2021). Proper assessment and management of project risks are helpful for software teams to increase the likelihood of overall project success (Buganová & Šimíčková, 2019).

The research literature includes many frameworks and models for assessing risks (Anes et al., 2020; Keil et al., 1998; Lunesu et al., 2021; Persson et al., 2009), with the simplest form being risk checklists (Boehm, 1991). While risk lists are easy to produce and use, they do not capture how risks can influence other risks, which entails that the impact of two risks can exceed the sum of the consequences of two individual risks (Eden et al., 2005; Williams et al., 1997). Researchers occupied with dependencies among risks in software projects still tend to use the traditional risk assessment methods, failing to acknowledge the implications of risks on other risks (Fang et al., 2012; Kazemi and Mosleh, 2012).

Managing risk dependencies in remains a paramount concern in contemporary software development that often relies on teams following agile principles such as "individuals and interactions over processes and tools" (Beck et al., 2001). As large scaled agile methods are adopted, dependencies between teams are created, leading to agile teams having to manage both dependencies inside and outside the team (Dikert et al., 2016). However, individuals may often not perceive the risks and their dependencies uniformly. Diverse perceptions may especially be present in agile teams, where ascribing to agile

principles can result in inconsistency issues (Sithambaram et al., 2021). Having diverse perceptions among software roles may cause them to perceive the necessary actions to address project risks differently, leading to conflict among the roles. Therefore, we propose that the roles in agile teams must examine their perceptual differences during risk assessment.

Despite the call for a broadened perspective of risks within software teams (Ackermann et al., 2007; Ackermann & Alexander, 2016), the perceptual differences among software roles in agile teams remain largely unexplored in the extant research. To address this research gap, we conducted an action case study (Braa and Vidgen, 1999; Vidgen and Braa, 1997) of an agile team in a large organization in the financial sector to answer the research question: *How can an agile team understand and manage different explanations of their software project risks?*

To study the different explanations in an agile team, we considered the roles of "Scrum Master," "Product Owner," "Developer," "Business Analyst," "UX Designer," and "Architect," using causal mapping (Ghobadi and Mathiassen, 2014; Laukkanen, 1994). Causal mapping is a technique for visualizing patterns of concepts and causal beliefs that reside in statements by different actors and groups. This technique allowed us to map the role-specific explanations of project risks in an agile team and assess the usefulness of these maps for the team in understanding and managing risks.

In the following, we take a starting point in the existing research on assessing software project risks in agile teams and causal mapping in the IS literature. Next, we outline the research approach, including our action case and how the data was collected and analyzed. The findings are then presented. Finally, we discuss how our findings contribute to previous research, their implications for practice, limitations, and the needed future research.

2 Existing literature

2.1 Assessment of Software Project Risks in Agile Teams

Traditionally, risk assessment is one of the core activities in risk management for software development projects. Risk assessment consists of three activities: risk identification, analysis, and prioritization. Risk assessment is concerned with documenting potential risks, assessing their likelihood of occurrence, and their potential harm to a team or project (Dingsøyr & Petit, 2021). However, extant literature is currently focused on risk assessment approaches in agile contexts (Odzaly et al., 2018) as the agile methodologies fail to promote concrete techniques and activities related to assessing project risks (Tavares et al., 2019, 2021). In agile teams, risk assessment is performed informally with a low amount of documentation, and they typically address risks through transparency, inspection, and adaptation (Schön et al., 2020; Sommerville, 2016). Although some researchers argue that this approach is inadequate, since the emergence of agile development, there have been calls for exploring formal assessment of risks to combat the lack of risk assessment in agile projects (Anes et al., 2020). Extant literature offers a wide range of models to assess risks (Persson et al., 2009; Suresh & Dillibabu, 2020) and specifically for agile teams (Anes et al., 2020; Lopes et al., 2021; Odzaly et al., 2018). The existing assessment models cover different types of risks, e.g., implementation risks (Lyytinen, 1987), requirements-related risks (Ramesh et al., 2010), distributed development risks (Persson et al., 2009), risks to effective knowledge sharing (Ghobadi and Mathiassen, 2016), and information security risks (Kuzminykh et al., 2021). Even though the previous models are diverse in nature, they all ignore potentially different causal explanations of project risks by different roles in an agile team.

2.2 Causal Mapping

Causal mapping is a type of cognitive mapping in which the units of analysis clarify their causal claims of specific concepts relating to a real-life situation. Causal mapping displays the patterns of concepts and causal beliefs in definite statements of different actors and groups. It is intended as a tool for

comparative analysis of different types of actors within an organization and as a means of identifying variations and similarities across the actors' perspectives (Laukkanen, 1994, 1998). The process of eliciting causal relationships is concerned with finding expressions that reveal concept A leads to concept B or B is an outcome of A. These causal assertions are visualized through models consisting of nodes and arrows linking the nodes. The nodes denote concepts and actors, such as project managers, perceive in their context. The arrows symbolize the actors' beliefs about causal relationships among the concepts (Laukkanen, 1994, 1998).

In the existing research, causal mapping has been used to show the consistency in perceived causal relationships of barriers to effective knowledge sharing across software roles in agile teams. In other words, causal mapping can visualize the barriers to effective knowledge sharing (Ghobadi and Mathiassen, 2014). Similarly, causal maps have been used to systematically evoke and represent users' perceptions of adopting a new system or technology (Ackermann et al., 2014; Ackermann and Alexander, 2016; Kjærgaard and Jensen, 2014; Williams et al., 1997). The causal maps can engage users to make their interpretations of a system concrete and facilitate reflection among users and managers to identify consistent perceptions that may inhibit the use and adoption of a system (Ackermann and Alexander, 2016; Kjærgaard and Jensen, 2014). Furthermore, causal mapping can act as a technique for practitioners and researchers to assess project risks by identifying and showcasing the causal relationships of risks (Al-Shehab et al., 2005; Hijazi et al., 2014). Utilizing causal mapping to assess risks can reduce the required resources for software projects (Hijazi et al., 2014), and the produced causal maps can be used as a tool for post-evaluation of past projects for software teams (Al-Shehab et al., 2004). Recently, the causal mapping technique has been used with fuzzy logic to assess risks in software projects. The researchers make use of complex mediating variables and mathematical equations to calculate the values of causal links between risks and the weight of the causal maps (Bakhtavar et al., 2021; Erbay & Özkan, 2018).

Extant research on causal mapping of project risks and their implications (Ackermann et al., 2014; Ackermann & Eden, 2020; Williams, 2017) adopts a team-specific view of project risks, concatenating all the perceptions of project risks into one generic causal map of a software team. To our knowledge, no one has used causal mapping to show role-specific explanations of project risks in an agile team, thereby adopting a role-specific view of project risks. In addition, prior research using causal mapping has a consensus of utilizing the case study approach (Ackermann et al., 2007; Ackermann and Alexander, 2016; Al-Shehab et al., 2006; Ghobadi and Mathiassen, 2014; Kjærgaard and Jensen, 2014), except for one literature review (Hijazi et al., 2014). To the best of our knowledge, causal mapping has not been used with a research focus on intervention in an agile software development practice.

3 Research approach

We followed an action case study approach, which consists of intervention from action research and interpretation from the case study approach (Braa and Vidgen, 1999; Vidgen and Braa, 1997). This approach is appropriate for investigating our combined practical and theoretical knowledge interest in how different roles in an agile team explain the causes of their project risks. To meet this knowledge interest, we elicited these risk explanations in cooperation with an agile team. Eliciting the risk explanations was an intervention, cf. action research (Iversen et al., 2004), as the explanations in the form of causal maps did not exist in the agile team before our investigation. The collaborative effort of the researchers and practitioners in creating causal maps occurred in the case of an agile software project team referred to as *Alpha* in a large Danish bank called *Estate Bank*. We selected the *Alpha* team as a paradigmatic case (Flyvbjerg, 2006) since they shared our interest in better understanding the causes of project risks and were open to exploring the usefulness of causal mapping for this purpose.

3.1 The Case of a SAFe Team in Estate Bank

Estate bank has 4000 employees and offers a wide range of products and services for banking and housing investments. Their main activities are mortgage credit and banking. The *Estate bank* is owned

by a Danish association of homeowners and corporations. *Estate Bank* employs approximately 400 employees developing software used in the organization and by other banks that offer their mortgage products. One hundred of these people work on a mortgage platform in a multi-team development effort, where team *Alpha*, which we collaborated with, is situated. The teams are organized according to the Scaled Agile Framework (SAFe) (*SAFe 5.0 Framework*, 2022), where a cluster of teams work towards shared goals and solutions. This is accomplished through the agile release train, which aims to deliver a continuous flow of value. The process begins with a fixed and reliable schedule, which is determined by the chosen rhythm of the program increment. Within each program increment, teams embark on a new system increment every two weeks, and all the teams are embedded in the same program increment, which lasts between 10-12 weeks. The program increments have standard start and end dates and duration. The teams working within a program increment must conduct the most important event in SAFe, the program-increment-planning event. The agenda of this planning event is to present the business context and vision, what is the most important to focus on and develop in the future, as well as identify risks to a program increment's success (Putta et al., 2018; *SAFe 5.0 Framework*, 2022).

Alpha consisted of 12 team members and included 1 Product Owner, 1 Scrum Master, 1 Business Analyst, 2 Architects, 4 Front-end Developers, 1 Back-end Developer, 2 UX Designers, and 1 Student Assistant. The team members in *Alpha* have worked together for a long time and know each other well, as several of the team members have worked in the team for over two years. Thus, the roles in *Alpha* form a closely connected team. Four of the twelve team members are in Poland, whereas the rest operate in Denmark. According to *Alpha*: The Product Owner's primary objective is to prioritize tasks in the backlog and to ensure that *Alpha* delivers the most critical tasks on time. The Scrum Master is responsible for ensuring adequate use of Scrum by following the Scrum guide, facilitating Scrum events, and shielding the team from impediments to sustain the focus of the team. The Business Analyst is the link between the IT department and the rest of the organization. The Business Analyst is responsible for defining the corporation's requirements and ensuring that the software solution's technical aspect lives up to the required quality. The UX designer analyzes the team's work cycle and the software requirements, defines requirements, and prepares tasks for the developers supported by design examples. The Front-end developers develop the front-end-related tasks and are also responsible for testing the software, whereas the Back-end developer is concerned with developing and maintaining API's. Finally, the Architect participates in developing both front-end and back-end related tasks and has a broader responsibility for the development process than the developers. Throughout this study, the Front-end and Back-end developer roles will be abstracted into one and referred to as Developer.

3.2 Data Collection and Analysis

This study relied primarily on interviews as the primary method of data collection. The first author conducted semi-structured interviews with 10 members from *Alpha* following an interview guide to ensure that the same questions were asked (Patton, 2015). The *Alpha* team members interviewed include 1 Product Owner, 1 Scrum Master, 1 Business Analyst, 2 UX designers, 2 Architects, and 3 Developers (cf. Table 1). In addition, the first author observed one of *Alpha's* retrospective meetings (1½ hours) and PI-planning meetings (6 hours), where *Alpha* discussed risks related to the team and the overall program. Field notes were conducted regarding risks and causal explanations, the mood, and the environment. These observations from the meetings supplement the interviews to achieve a higher level of data triangulation (Host et al., 2012). The same applies to historical data, consisting of documentation of *Alpha's* 10 latest retrospective meetings and the team and program-related risks from the three latest PI-planning meetings to obtain an even deeper understanding of the situation and context.

Role	Initial interview	Validation interview
Scrum Master	(46:37 min)	(29:33 min)
Product Owner	(37:06 min)	(19:55 min)
Business Analyst	(50:42 min)	(29:19 min)

UX Designer	(30:42 min) (50:05 min)	(23:55 min)
Architect	(01:04:04 min) (50:10 min)	(35:55 min)
Developer	(46:41 min) (32:09 min) (56:21 min)	(18:48 min)

Table 1: Overview of interviews

The data analysis was initiated by reviewing the field notes and the recordings generated from the interviews and transcribing the parts of the interviews mentioning risks and causal relationships. The first author coded statements from the transcripts and the field notes to uncover the roles' identified risks (which produced a risk list for each role) and the causal explanations for each risk. A concern mentioned by the interviewees was identified as a risk when it corresponded to the definition of a project risk (Pressman and Maxim, 2015; Sommerville, 2016), which is an issue that potentially threatens the project plan. Examples of project risks include "lack of staff," "unrealistic budget," or "exceeded schedule" (Pressman & Maxim, 2015). We solely identified risks and their causal links. We did not include any mediating variables, such as risk causes or risk root causes. A causal relationship between two or more risks was identified when a risk was mentioned as a contributor to the occurrence of another risk. When the causal explanations were analyzed and the causal relationships identified, the construction of the causal maps began. All the identified risks included in the causal maps were mentioned by the interviewees themselves, whereas a few causal links were solely identified by the researchers. The causal mapping approach was selected as the analytical framework since it provides a versatile approach for explanatory cognitive studies and presents the researcher with a more practical and powerful tool to analyze data and concepts than a text-based analysis (Laukkanen, 1998; Laukkanen and Eriksson, 2013).

The identified risks were further categorized into actor, structure, task, and technology (Lyytinen et al., 1998). An actor-related risk pertains to individuals and or groups of stakeholders, such as software roles or users. Structural risks are the structures within an organization, e.g., structures of authority and workflow. Task risks are implications of specific tasks, such as task uncertainty or ambiguity of task descriptions. Technological risks are tools, methods, and infrastructure used to develop software (Lyytinen et al., 1998). This categorization ensured comprehensive attention shaping, as suggested in previous risk management research (Persson and Schlichter, 2015), and an overview of the identified risks. In Figure 1, we summarize the number of risks and causal links in the four socio-technical categories we identified from the interviews with *Alpha*. Figure 1 shows that *Alpha* is mostly concerned with the causal implications from actor-related risks to other actor-related risks (23) and second to task-related risks (21). *Alpha* is the least concerned with the causal implications of the technology-related risks (0, 2, 2, 4).

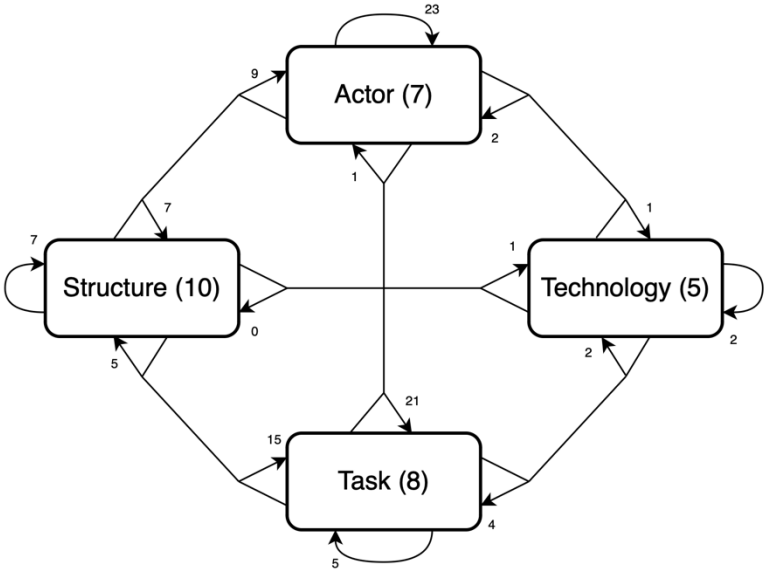

Figure 1: Total number of identified risks and causal links between each socio-technical category.

We validated the role-specific causal maps with representatives from each role in *Alpha* (cf. the validation interview column in Table 1). These causal maps were constructed through two iterations. First, the researchers constructed a causal map for each role on account of the statements from the interviews, the participatory observations, and the historical data. Second, the causal maps were adjusted based on the feedback from each role representative. Only minor adjustments were identified during validation and the causal links solely identified by the researchers were either discarded or confirmed by the participants. A workshop followed the individual feedback sessions with *Alpha*, where the causal maps were presented to all the team members from *Alpha*. The workshop provided feedback on the appropriateness of the causal maps, the roles' reflections on their diverging causal explanations of project risks, and their usefulness for *Alpha*.

4 Findings

In the following, we present the causal maps for each of the six roles. Before presenting the findings for each role, we will briefly summarize the main takeaways from *Alpha* to ensure adequate comprehensibility of the findings. The different roles were primarily concerned with the risk of "missing their deadlines" but differed in their causal explanations. The developer emphasized "technical debt" as a direct causal explanation for missing their deadlines, with two other technological risks as indirect causal explanations, "developing insufficient code" and "inadequate maintenance of open-source library." The architect also identified technological risks as contributing causal explanations to missing their deadlines, although these were more generic as the architect is responsible for the overall development decisions. The architect and developer were the only ones to identify technological risks.

The UX designer mentioned the risk of "insufficient design" and "insufficient balance between tasks" as unique causal explanations for missing their deadlines. At the same time, the Business Analyst and the developer were the only roles to identify the risk of "insufficient communication" as a causal explanation for missing their deadlines. The Business Analyst attributed communication difficulties across teams, where the developer experiences the difficulties across locations. Part of the developers' work is communicating with the developers stationed in Poland, and the Business Analyst is responsible for communicating with the other teams in the program. The Business Analyst, acting as a bridge between the *Estate bank* and *Alpha*, was also the only one to perceive insufficient communication as the primary explanation for missing their deadlines. Thus, the different causal explanations reflect the function and focus of the specific roles.

The Product Owner was the only role to identify the risk of "unanticipated complexity in development" and as a part of the causal explanation for missing their deadlines. This unanticipated complexity could be the specific technology-related risks identified by the developer and the architect, which resonates with the Product Owner's general view of the team's development process. The Product Owner does not necessarily need to know the complexity, as it is not part of their role's responsibility.

The Scrum Master did not perceive the risk of missing their deadline as a risk worth focusing on and did not provide any causal explanations for the risk. The Scrum Master was instead mainly worried about the team becoming too large. In the following, we will present the causal maps and explanations for each of the six roles. Note that the underlined nodes in the maps entail that the risks were perceived as the most important risks by the specific role.

4.1 Developer

To explain the concern of missing their deadlines, a developer stated, "*Management might pressure us to do the delivery faster, even though they know it is probably not possible."* Moreover:" *Once a date is set, you have to meet it because other teams are dependent on that... the date becomes real at some point*". These statements show that one explanation for missing their deadlines is management's planning of their deliveries.

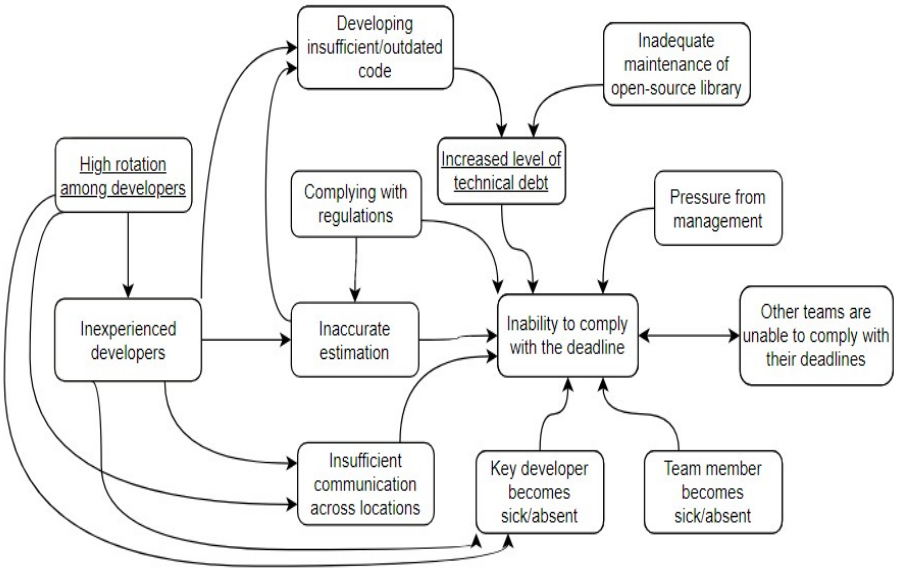

Figure 2: Developer causal map

The risk of missing deadlines can be inscribed at the beginning of every program increment. The developer also mentioned new regulations impacting what they are working on and how long production will take, thus contributing to inaccurate task estimation and *Alpha* missing a deadline. The rationale was that it might take up unexpected time from some of the team members. The inaccurate estimation of tasks "*could have the consequence of having spillovers, where the tasks pass over to the next sprint*." And was seen as a contributing factor in explaining delayed tasks. A Danish developer explained the insufficient communication across locations' causal relationship with the risk as a difference in work schedule and culture "*If I, for example, have a task I find difficult, then I would much rather go to the team members sitting in this office."* Showing a hesitation in communication or asking for help from a team member across locations. The developer emphasized technical debt as an explanation for missing a deadline:" Technical *debt significantly impacts meeting our deadlines, because if you generate a lot of technical debt, it will take longer to finish a task, and then it is not likely you will meet your deadline."*

The developer continued to highlight the risk of technical debt as one of the most essential risks in the causal maps, as shown in Figure 2 with an underscore. Technical debt was explained to be caused by technology-related risks, for instance: "*We currently have some outdated code, and I know there are tasks in the next PFP (meeting), which are affected by this code, and I know, due to this technical debt from this outdated code that the task will take longer to finish in the future if we do not fix it."* And: "*One of the examples of technical debt is actually having these open-source libraries or other dependencies that do not get maintained."* Thus, he sees outdated or otherwise insufficient code, unexpected issues with open-source libraries, and the inherent accumulation of technical debt impacting the likelihood of meeting their deadlines. Even though the developer and architect share a similar emphasis on technological risks, they differ in terms of the identified technological risks and some of their causal explanations.

4.2 Architect

The architect did not identify "inaccurate estimation" as a causal explanation for missing their deadlines. This difference can be due to the developers being mainly responsible for the estimations. Unlike the developer, the architect explained the implications of complying with regulations: "*Then you just have to react immediately. They are so hard to see when they come. You have to take care of it when it presents itself. It can push to stuff, it really can".* The sudden allocation of human resources to deal with regulations imposed by the Danish government may result in rescheduling other tasks currently in their backlog. Complying with regulations was explained to be of higher priority than many other tasks, which can ultimately lead to a failure to adhere to the deadline. This clarifies the connection between the two risks in Figure 3. The architect was worried about inadequate maintenance of open-source libraries, similar to the developer. Still, the architect did not present a causal explanation that made it possible to

deduce an apparent connection to any other identified risks. The architect did not explain its implications for increasing technical debt, as the developer but pointed to inadequate maintenance of libraries as causing problems for their project: "*It took us a long time to figure out what we should do about it. Should we choose another one and then migrate all of our applications over to this or should we take the lead and help this library on the way?".* The risk has implications for the risk of not meeting their deadline, should they suddenly spend additional resources on resolving the issue of inadequate maintenance of libraries.

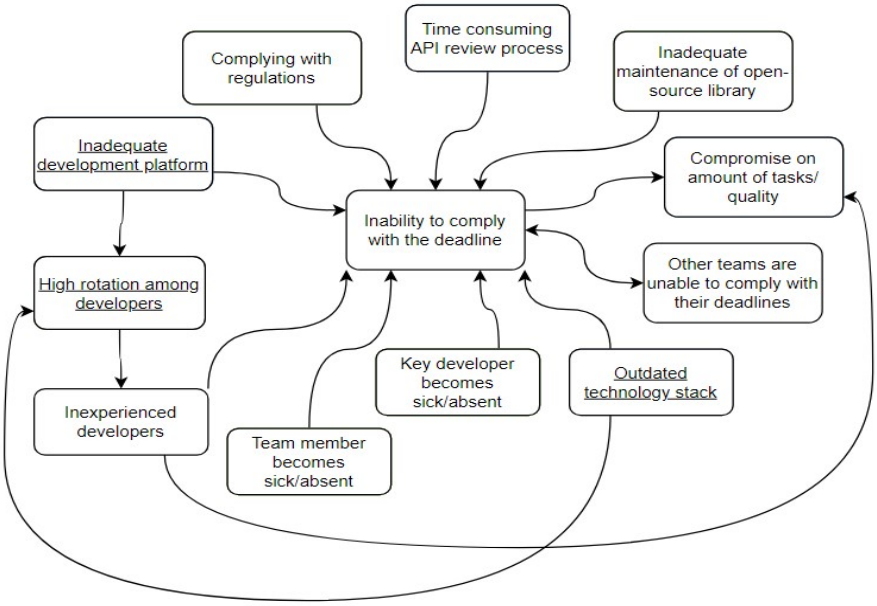

Figure 3: Architect causal map

The architect highlighted the importance of two technological risks," Inadequate maintenance of development platform" and" outdated technology stack." He stated: "*I want the tools that make me effective. Otherwise, I'll get very irritated. It just makes me happy, having proper tools that just work".* The architect clarified that they have implications for the high rotation: "*Some of these things may be interrelated, e.g., this technology stack may constitute a higher rotation among developers because they are working with something they really do not want to."* This explains the connection with the risk of high rotation and adds another layer behind the explanation to the high rotation among developers. The risk was also presented to be caused by technological issues, besides the unsatisfactory salary for the developers in Poland. The two former roles place a significant emphasis on technological risks and explanations related to development, while the UX designer emphasizes structure-related explanations and risks essential for the role.

4.3 UX designer

Similar to the developer, the UX designer perceived "inaccurate estimation" as an explanation for their inability to meet their deadlines. The UX designer explained the causal relationship after being asked about the causes of delays: "*, there are many, but the typical risks that we meet can, for instance, be that someone has misestimated a task. It has been estimated that something is easy, and it turns out to be difficult".* Unlike the developer, who viewed inaccurate estimations as unimportant due to how infrequent they occur. The UX designer identified inaccurate estimations as a typical cause of delays and did not mention it as something that happens infrequently. This alludes to a slight inconsistency in the risk perception between the developer and the UX designer. The designer was the only role to mention the risk of lacking clarification from enterprise architects, change in prioritization, the insufficient balance between tasks, and the two design-related risks. As shown in Figure 4, all the risks can be ascribed to causing an impact on *Alpha's* ability to comply with their deadlines because they require *Alpha* to spend an additional amount of time and resources if the risks occur. The researcher

reasoned that the risk of an insufficient design and balance leads directly to missing deadlines. The three other risks were modeled on account of the previously presented explanations to have an indirect impact on meeting their deadlines as they contribute to the occurrence of insufficient design and insufficient balance between tasks. The UX designer's unique identification of risks and explanations related to analysis and design can be due to the role's responsibility. In contrast to the developer and the architect, the UX designer mentioned no technological risks and separated themselves from the other roles by identifying design and analysis-related explanations. Although the UX designer did not identify "insufficient communication" as a significant risk or causal implication, the next role perceived it as the most important.

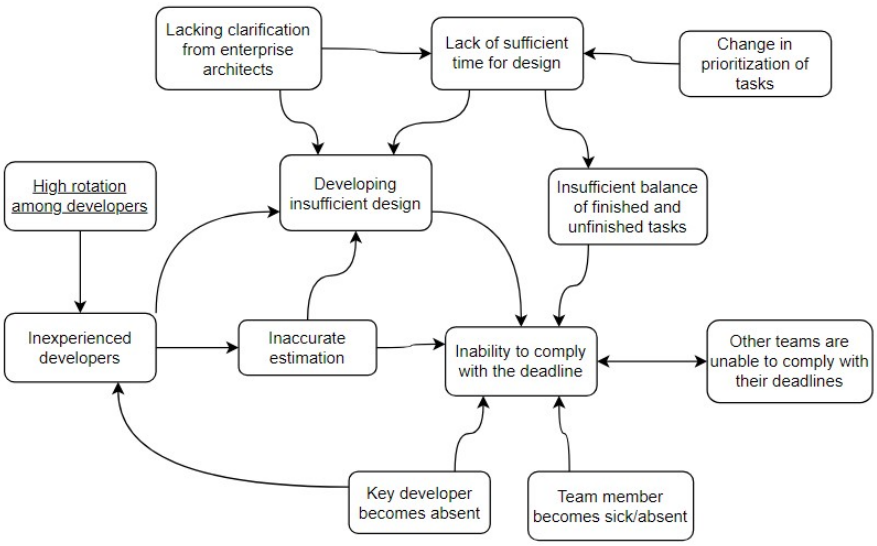

Figure 4: UX designer causal map

4.4 Business Analyst

The Business Analyst identified insufficient communication as the most critical risk, and according to the Business Analyst: "*For me in my position, then it is usually lacking communication or the lack of listening to communication that causes many of these problems."* As shown in Figure 5, insufficient communication leads to three risks in the causal map, but according to the Business Analyst, it can contribute to many other risks. This signifies the Business Analyst's unique perception of insufficient communication compared to the other roles in *Alpha*, which can be explained aside from the fact that communication was a central part of the role's responsibility. Communication can be ingrained in the role itself: "*I have a background in communication, and as far as I can see, that is usually why something goes wrong."* The role referred to their educational background, which further contributes to their personal bias in perceiving communication issues as the root of most of *Alpha's* concerns.

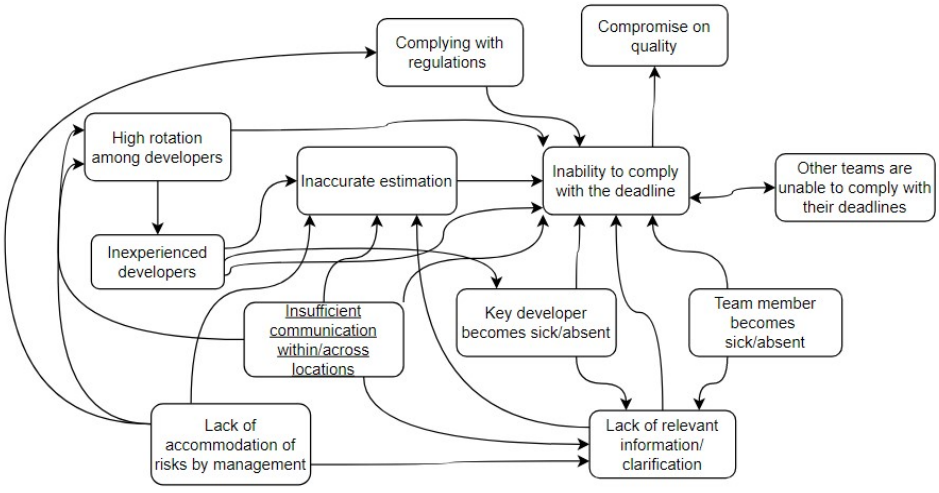

Figure 5: Business Analyst causal map

The Business Analyst perceived the communication issue to primarily be across the different teams in the program. The role did not share a similar emphasis on insufficient communication across locations within the team with the developer. It was revealed: "*I often have issues with communication with other teams; if I want a quick answer on something, then I'm told to send a carrier pigeon."* The Business Analyst clarified that he could end up waiting a week for an answer that should have taken two minutes and: "*It has been communication problems that have caused things not to be delivered on time or have a high enough quality."* This showed that the Business Analyst perceives poor communication as a risk with significant consequences for delivering on time and further supports the connection with the risk of lacking clarification/information. This difference in focus in terms of insufficient communication can be because of the difference in responsibility and who the developer and the Business Analyst primarily communicate with. The Business Analyst acts as the link between the corporation's wishes and the team, which entails a lot of communication with the other teams within the SAFe program. Whereas the developer works closely with the Polish developers seated in Poland, the Business Analyst might not be as dependent on communicating with them. The Business Analyst did not experience the same issues with communication across locations when asked: "*A little bit; I will say we are lucky in my team that we have been together for so many years, so I will say we are good at talking with each other in the team and be transparent."* This difference in who both roles mainly communicate with can explain the variation in their experience of insufficient communication. The next role differentiates from the former by presenting risks and causal explanations that were not mentioned by any other roles.

4.5 Scrum Master

The Scrum Master differentiates from the other roles by being the only one who did not identify the risk of missing deadlines and getting delayed. This is due to the: "*We just take less into the Program Increment. Then management may have set some deadlines, they must move, but it is not something we experience as a team."* The Scrum Master differentiates from the other roles by not perceiving delays as a concern for the team. The Scrum Master was primarily concerned with their team becoming too big. The Scrum Master explained the cause of potentially becoming too big in the future: "*In my opinion, it is caused by the fact that there are so many tasks on our table… we can see many tasks are coming our way". This emphasizes there is currently an overflow of tasks that Alpha must take care of and*: "*Therefore, we need to take in more developers into the team to solve it."* The overflow of tasks entails *Alpha* to increase their number of developers in the team, thereby potentially becoming too big. At the same time: "*The more you put in, the more tasks you receive."* Thus, the Scrum Master argues that an increase in the team's size can lead to a rise in the number of tasks. *Alpha* is the only front-end team in their program, which means all the front-end tasks must be solved solely by *Alpha* and increase the number of developers in the team. The Scrum Master shares few similar causal explanations with the other roles, but the role shares the fewest with the Product Owner.

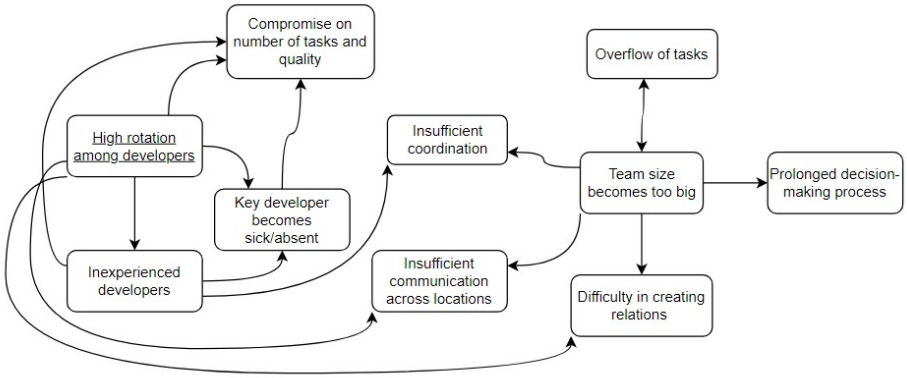

Figure 6: Scrum Master causal map

4.6 Product Owner

The Product Owner has a reversed perception of the causal relationship between technical debt and meeting deadlines compared to the developer. The Product Owner explained: "*When we are under pressure, we must make cuts somewhere, and deadlines are typically the most important, so it is often just technical debt we build up. This is what we are seeing now.".* The Product Owner perceives missing their deadlines as a cause of increasing their technical debt instead of the causal relationship being the

other way around. This perception contrasts with the developer perceiving the generation of technical debt as one of the major causes of missing deadlines. The two roles agree fundamentally on the same connection between risks. Still, the developer was arguably more concerned with the increase of technical debt causing delays on the project sometime in the future. The Product Owner focused on the nearest deadline; if they were to miss their deadline potentially, they must accept generating technical debt to meet their forthcoming deadline.

The Product Owner was the only role to identify the risk of unanticipated complexity in development, which seemed more like a risk that the developer or architect would have recognized because of its technical nature. The Product Owner stated: "*There is always this risk of some tasks being more complex. It will cause us not to meet our deadlines, or the alternative is that we build up more technical debt".* This unexpected complexity can lead to a task taking longer to complete than anticipated, which causes *Alpha* to be unable to comply with their deadlines or accumulate more technical debt instead. When looking at the Product Owner's causal map and comparing it to the developer and the architect's map, some similarities across the maps materialize. The developer explained that the generation of technical debt was caused by redundant or insufficient code and a lack of maintenance of open-source libraries. The architect identified the inadequate development platform, outdated technology stack, lack of maintenance of libraries, and the time-consuming API review process as contributing factors to their inability to comply with their deadlines. These risks were related to development, which implies that the unexpected complexity in development the Product Owner identified is an aggregation of the technological risks identified by the developer and the architect.

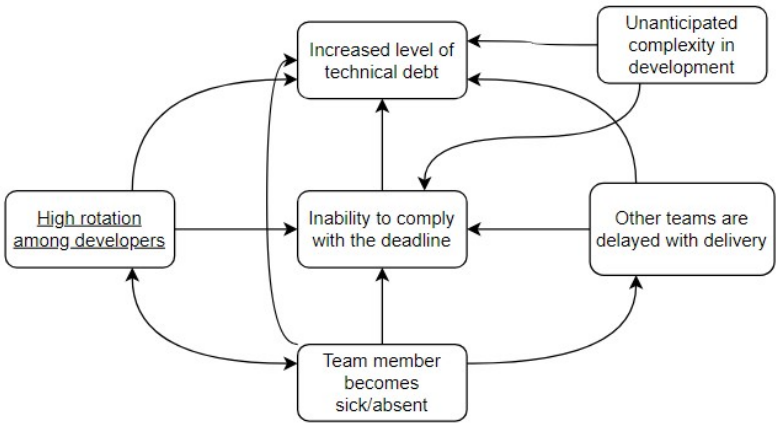

Figure 7: Product Owner causal map

4.7 Workshop: Team reflections on the causal maps

Alpha found the causal maps interesting and reflected on what they learned from the maps. One of the developers mentioned that it was interesting to see the differences between the identified risks among the roles. The same developer clarified why it was interesting: "*How we see some risks differently because they directly touch one role and not necessarily another."* Where the developer found the substantial differences between the roles interesting, the Business Analyst was interested in another difference: "*I also think it's interesting to see the different levels of the same type of risks, as you mentioned this sort of abstraction."* The statements from the developer and the Business Analyst show that the different roles in *Alpha* have broadened their perspective and gained insight new into each other's limitations and differences in terms of causal explanations of risks.

A new developer in *Alpha* who previously worked at a different team within *Estate Bank* explained: "*I agree on all of them. The only one I have not met before is, of course, the open-source library. I think that is a Front-end thing. But the rest of it, I think you would find that most of the teams are hit by*". The new developer stated that the identified risks could also exist in the other teams in *Estate Bank*, except for the risk related to open-source libraries, which was a unique concern for a front-end team. This implies the usefulness of *Alpha's* insights from the maps and the value of using them as a knowledge-sharing tool. Several roles identified the use of the causal maps as a tool for knowledge sharing within *Alpha*; a designer mentioned: "*Yeah, that was my thought too that this could be used as an input to a risk management discussion."* When addressing risks, *Alpha* can use causal maps as a starting point. They can restrict the most severe risk and which they should attempt to address. The Scrum Master

explained more specifically: "*But also, we could use it for a retrospective and talk about it inside the team."* A UX designer reflected on the causal maps' use in more general terms concerning risk assessment. In contrast, the Scrum Master identified the applicability of the causal maps in relation to future sprint retrospective meetings.

An architect continued to elaborate on using the causal maps across teams as a knowledge-sharing and documentation tool for their PI-planning meetings. An architect explained: "*I also think that this whole root causes analysis of these impediments that we have identified, where you can use this map to identify, which are connected, and that's also a thing we haven't been very skilled at doing and we have stopped doing it for several PFPS."* The architect revealed that the causal maps could be valuable in providing the different teams with an overview of the interconnectedness between the various risks instead of only considering risks in isolation. The designer continued to clarify the use of the maps as a knowledge-sharing tool: "*I also think this historical perspective can help warrant and investigate in handling a risk, so if we see the same risk over and over again, it should be an argument that we can spend some money and time on addressing the risk."* These statements highlight the maps' usefulness for creating actionable knowledge about project risks in the agile team.

5 Discussion

Our action case study of understanding and managing software project risks, as seen by different roles in the agile team *Alpha* at *Estate Bank*, has two key findings:

First, our action case study shows that *causal mapping is useful for revealing role-specific explanations of software project risks in agile teams*. Numerous risk assessment tools and models of software project risks have been proposed recently, but many still fail to consider the mutual implications of risks (Lopes et al., 2021; Suresh & Dillibabu, 2020; Tavares et al., 2021). This study extends previous research that found causal mapping can unfold the mutual implications of risks (Ackermann et al., 2014; Ackermann & Eden, 2020; Williams, 2017). While the extant research using causal mapping has typically adopted a team-level perspective when showcasing the perceived implications of project risks (Ackermann & Alexander, 2016; Al-Shehab et al., 2006; Hijazi et al., 2014; Williams, 2017). We add to the team-level perspective by providing role-specific causal maps showcasing the implications of project risks for each role in an agile team. Adopting a role-specific perspective preserves each role's nuances instead of leaving them out in a team-level perspective. *Alpha* is a very composed team, where several of the team members have worked together for over two years. The role-specific causal maps can reveal the heterogeneities of a software team, regardless of how homogenous the team seems on the surface.

Second, our action case shows that *agile teams can use role-specific causal maps to juxtapose their explanations of software project risks.* The insights gathered from each role provide knowledge into distinctive perceptual differences that exist during agile development, where committing to agile values and principles may intensify issues related to inconsistencies (Anes et al., 2020; Pikkarainen et al., 2008). Our insights from section 4 align with existing research connecting roles to the tasks they perform (Huisman and Iivari, 2006). All the roles have identified project risks and implications unique to their role, highlighting inconsistency in the team. For instance, how the developers and architects emphasize technological implications more than the other roles that are not directly involved with coding. These inconsistencies and perceptual differences among the roles are inescapable in an agile team. Thus knowledge of how they materialize in agile teams discloses new insights into the IS research focusing on the interaction between roles (Ghobadi & Mathiassen, 2014, 2016)

Our findings have implications for practice, as we show how causal mapping may be useful for agile teams. Should the technique be adopted by practitioners, it will be beneficial to assign the responsibility of constructing the causal maps to the Scrum Master. The Scrum Master is responsible for facilitating the agile team's Scrum events, and this person could use the maps in retrospective meetings. Although, the Scrum Master should be aware of the time-consuming process of using and constructing causal maps, as pointed out in previous research on the technique (Kjærgaard and Jensen, 2014).

This study has limitations regarding the creation of causal maps. The different roles from *Alpha* constructed their subjective perception of risks and their causal implications. The constructed causal maps signified a socially contrived perspective of how different roles in a software team perceived project risk. They also represented a subjective explanation of risks created through the interaction between the role owners and the researchers. This limitation of causal mapping falling subject to the subjectivity of the person constructing the map may undermine its trustworthiness, which aligns with prior research on causal mapping (Kjærgaard and Jensen, 2014). In this study, the credibility of the causal maps was ensured by conducting individual and collective validation with *Alpha*.

Future research on causal mapping for software project risks may attend to longitudinal studies regarding agile teams' perceptions of project risks and causal explanations. While the causal maps in our study were static snapshots of *Alpha's* explanations of project risks, the state of some risks and their explanations will likely change over time. Other researchers could also investigate the practical usefulness of the causal maps for software teams of different sizes, and which issues they encounter. Longitudinal action research studies could further investigate whether the resources spent on constructing the causal maps are proportional to the value it provides for an agile team.

6 Conclusion

We employed causal mapping in an action case research effort to examine how roles in an agile team explain software project risks. The investigated case was an agile team with six roles: Developer, Architect, UX designer, Business Analyst, Scrum Master, and Product Owner. We produced six role-specific causal maps from interviews with these roles and then validated them with each role. We then presented the role-specific causal maps to the team in a workshop. The juxtaposing of these maps revealed unique insights into the different roles they were unaware of, and the causal maps were perceived as useful for the agile team. With this action case study, we document the usefulness of idiosyncratic risk explanations as a contribution to the software project risks literature.

References

Ackermann, F., & Alexander, J. (2016). Researching complex projects: Using causal mapping to take a systems perspective. *International Journal of Project Management*, *34*(6), 891-901.

Ackermann, F., & Eden, C. (2020). Strategic Options Development and Analysis. In M. Reynolds & S. Holwell (Eds.), *Systems Approaches to Making Change: A Practical Guide*, 139–199.

Ackermann, F., Eden, C., Williams, T., & Howick, S. (2007). Systemic Risk Assessment: A Case Study. *The Journal of the Operational Research Society*, *58*(1), 39–51.

Ackermann, F., Howick, S., Quigley, J., Walls, L., & Houghton, T. (2014). Systemic risk elicitation: Using causal maps to engage stakeholders and build a comprehensive view of risks. *European Journal of Operational Research*, *238*(1), 290–299.

Al-Shehab, A., Hughes, R., & Winstanley, G. (2004). Using causal mapping methods to identify and analyse risk in information system projects as a post-evaluation process. *11th European Conference on Information Technology Evaluation*, 9-16.

Al-Shehab, A., Hughes, R., & Winstanley, G. (2006). CorMod: A Causal Mapping Approach to Identifying Project Development Risk. *European and Mediterranean Conference on Information Systems*, 6-7.

Al-Shehab, A. J., Hughes, R. T., & Winstanley, G. (2005). Facilitating Organisational Learning Through Causal Mapping Techniques in IS/IT Project Risk Management. *Professional Knowledge Management*, 145-154.

Anes, V., Abreu, A., & Santos, R. (2020). A New Risk Assessment Approach for Agile Projects. *International Young Engineers Forum*, 67–72.

Bakhtavar, E., Valipour, M., Yousefi, S., Sadiq, R., & Hewage, K. (2021). Fuzzy cognitive maps in systems risk analysis: A comprehensive review. *Complex & Intelligent Systems*, *7*(2), 621–637.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., & Thomas, D. (2001). *Manifesto for Agile Software Development*.

Boehm, B. W. (1991). Software risk management: Principles and practices. *IEEE Software*, *8*(1), 32-41.

Braa, K., & Vidgen, R. (1999). Interpretation, intervention, and reduction in the organizational laboratory: A framework for in-context information system research. *Accounting, Management and Information Technologies*, *9*(1), 25-47.

Buganová, K., & Šimíčková, J. (2019). Risk management in traditional and agile project management. *Transportation Research Procedia*, *40*, 986–993.

Dikert, K., Paasivaara, M., & Lassenius, C. (2016). Challenges and success factors for large-scale agile transformations: A systematic literature review. *Journal of Systems and Software*, *119*, 87–108.

Dingsøyr, T., & Petit, Y. (2021). Managing layers of risk: Uncertainty in large development programs combining agile software development and traditional project management, *De Gryuter*, 75-96.

Eden, C., Ackermann, F., & Williams, T. (2005). The Amoebic Growth of Project Costs. *Project Management Journal*, *36*(2), 15–27.

Erbay, B., & Özkan, C. (2018). Fuzzy FMEA Application Combined with Fuzzy Cognitive Maps to Manage the Risks of a Software Project. *European Journal of Engineering and Formal Sciences*, *2*, 6–21.

Fang, C., Marle, F., Zio, E., & Bocquet, J.-C. (2012). Network theory-based analysis of risk interactions in large engineering projects. *Reliability Engineering & System Safety*, *106*, 1–10.

Flyvbjerg, B. (2006). Five Misunderstandings About Case-Study Research. *Qualitative Inquiry*, *12*(2), 219-245.

Ghobadi, S., & Mathiassen, L. (2014). Perceived barriers to effective knowledge sharing in agile software teams. *Information Systems Journal*, 26(2), 95-125.

Ghobadi, S., & Mathiassen, L. (2016). Risks to Effective Knowledge Sharing in Agile Software Teams: A Model for Assessing and Mitigating Risks. *Information Systems Journal*, *27(6), 699-731*.

Hauck, J. C. R., & Vieira, M. (2021). Towards a Guide for Risk Management Integration in Agile Software Projects. In M. Yilmaz, P. Clarke, R. Messnarz, & M. Reiner (Eds.), *Systems, Software and Services Process Improvement*. Springer International Publishing, 73-87.

Hijazi, H., Alqrainy, S., Muaidi, H., & Khdour, T. (2014). Identifying Causality Relation between Software Projects Risk Factors. *International Journal of Software Engineering and Its Applications*, 8(2), 51-58.

Host, M., Rainer, A., Runeson, P., Regnell, B., Regnell, B., & Regnell, B. (2012). *Case Study Research in Software Engineering: Guidelines and Examples*. John Wiley & Sons, Incorporated.

Huisman, M., & Iivari, J. (2006). Deployment of systems development methodologies: Perceptual congruence between IS managers and systems developers. *Information & Management*, *43*(1), 29–49.

Iversen, J. H., Mathiassen, L., & Nielsen, P. A. (2004). Managing Risk in Software Process Improvement: An Action Research Approach. *MIS Quarterly*, *28*(3), 395–433.

Kazemi, R., & Mosleh, A. (2012). Improving Default Risk Prediction Using Bayesian Model Uncertainty Techniques. *Risk Analysis*, *32*(11), 1888–1900.

Keil, M., Cule, P. E., Lyytinen, K., & Schmidt, R. C. (1998). A framework for identifying software project risks. *Communications of the ACM*, *41*(11), 76–83.

Kjærgaard, A. L., & Jensen, T. B. (2014). Using Cognitive Mapping to Represent and Share Users' Interpretations of Technology. *Communications of the Association for Information Systems*, *34(1), 1097-1114*.

Kuzminykh, I., Ghita, B., Sokolov, V., & Bakhshi, T. (2021). Information Security Risk Assessment. *Encyclopedia*, *1*(3), 602-617.

Laukkanen, M. (1994). Comparative Cause Mapping of Organizational Cognitions. *Organization Science*, *5*(3), 322-343.

Laukkanen, M. (1998). Conducting Causal Mapping Research: Opportunities and Challenges. *Managerial and Organizational Cognition*, 168-189.

Laukkanen, M., & Eriksson, P. (2013). New designs and software for cognitive causal mapping. *Qualitative Research in Organizations and Management*, *8*(2), 122-147.

Lopes, S., Souza, R., Contessoto, A., & Oliveira, A. (2021). *A Risk Management Framework for Scrum Projects*, 30–40.

Lunesu, M. I., Tonelli, R., Marchesi, L., & Marchesi, M. (2021). Assessing the Risk of Software Development in Agile Methodologies Using Simulation. *IEEE Access*, *9*, 134240–134258.

Lyytinen, K. (1987). Different perspectives on information systems: Problems and solutions. *ACM Computing Surveys*, *19*(1), 5–46.

Lyytinen, K., Mathiassen, L., & Ropponen, J. (1998). Attention Shaping and Software Risk—A Categorical Analysis of Four Classical Risk Management Approaches. *Information Systems Research*, *9*(3), 233-255.

Odzaly, E. E., Greer, D., & Stewart, D. (2018). Agile risk management using software agents. *Journal of Ambient Intelligence and Humanized Computing*, *9*(3), 823-841.

Patton, M. Q. (2015). *Qualitative research & evaluation methods: Integrating theory and practice* (Fourth edition). SAGE Publications, Inc.

Persson, J. S., Mathiassen, L., Boeg, J., Madsen, T. S., & Steinson, F. (2009). Managing Risks in Distributed Software Projects: An Integrative Framework. *IEEE Transactions on Engineering Management*, *56*(3), 508–532.

Persson, J., & Schlichter, B. (2015). Managing Risk Areas in Software Development Offshoring: A CMMI Level 5 Case. *Journal of Information Technology Theory and Application (JITTA)*, *16*(1).

Pikkarainen, M., Haikara, J., Salo, O., Abrahamsson, P., & Still, J. (2008). The impact of agile practices on communication in software development. *Empirical Software Engineering*, *13*(3), 303–337.

Pressman, R. S., & Maxim, B. R. (2015). *Software engineering: A practitioner's approach* (Eighth edition). McGraw-Hill Education.

Putta, A., Paasivaara, M., & Lassenius, C. (2018). Benefits and Challenges of Adopting the Scaled Agile Framework (SAFe): Preliminary Results from a Multivocal Literature Review. *Product-Focused Software Process Improvement*. Springer International Publishing, 334–351.

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

SAFe 5.0 Framework. (2022). Scaled Agile Framework. https://www.scaledagileframework.com/

Schön, E.-M., Radtke, D., & Jordan, C. (2020). Improving Risk Management in a Scaled Agile Environment. In V. Stray, R. Hoda, M. Paasivaara, & P. Kruchten (Eds.), *Agile Processes in Software Engineering and Extreme Programming*. Springer International Publishing, 383, 132–141.

Sithambaram, J., Nasir, M. H. N. B. M., & Ahmad, R. (2021). Issues and challenges impacting the successful management of agile-hybrid projects: A grounded theory approach. *International Journal of Project Management*, *39*(5), 474–495.

Sommerville, I. (2016). *Software engineering* (10. ed., global ed). Pearson.

Suresh, K., & Dillibabu, R. (2020). A novel fuzzy mechanism for risk assessment in software projects. *Soft Computing*, *24*(3), 1683–1705.

Tavares, B. G., da Silva, C. E. S., & de Souza, A. D. (2019). Risk management analysis in Scrum software projects. *International Transactions in Operational Research*, *26*(5), 1885-1904.

Tavares, B. G., Keil, M., Sanches da Silva, C. E., & de Souza, A. D. (2021). A Risk Management Tool for Agile Software Development. *Journal of Computer Information Systems*, *61*(6), 561-570.

Vidgen, R., & Braa, K. (1997). Balancing Interpretation and Intervention in Information System Research: The Action Case Approach. *International Conference on Information Systems and Qualitative Research, Philadelphia, Pennsylvania, USA,* 524–541.

Williams, T. (2017). The Nature of Risk in Complex Projects. *Project Management Journal*, *48*(4), 55–66.

Williams, T., Ackermann, F., & Eden, C. (2013). Project risk: Systemicity, cause mapping and a scenario approach. In K. A. Artto & K. Kahkonen, (Eds.), *Managing Risks in Projects*, 343–352.