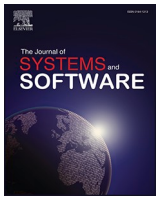
The Journal of Systems and Software 211 (2024) 111966

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss



Shedding light on the dark side – A systematic literature review of the issues in agile software development methodology use

Jan-Niklas Meckenstock

Chair of Industrial Information Systems, University of Bamberg, An der Weberei 5, Bamberg 96047, Germany

ARTICLE INFO

Editor: Liu Xiao

Keywords:
Agile software development methodologies
Issues in ASD
Dark side of ASD
Information systems development agility
Systematic Literature Review

ABSTRACT

Context: Agile software development (ASD) methodologies address problems of traditional approaches, but can also cause various issues. A systematization of the spectrum of negative facets and how they arise has not been developed, leaving the dark side of ASD opaque.
Objective: The paper systematizes findings on various negative facets to define *what* specific issues constitute the dark side of ASD. It also examines *how* relationships among these issues define the complexity of the downside.
Method: A literature review of 70 articles was conducted following Kitchenham & Charters (2007). The content analysis employed the coding procedure by Gioia et al. (2013) to systematize the negative aspects of ASD methodologies.
Results: The paper reveals 90 issues, grouped into 18 coherent themes. Six levels of manifestation point to the primarily affected constituents in ASD projects. In addition, 65 relationships among issues are explored, providing explanations for the complexity of the dark side of ASD.
Conclusion: The systematization shows *what* aspects constitute the dark side of ASD, emphasizing its multidimensional nature along issues such as reduced developer well-being, product quality and development productivity. The analysis of *how* its complexity is defined reveals that customer misbehavior and delivery pressure are significant origins of other issues.

1. Introduction

Agile software development (ASD) methodologies have gained considerable popularity in software development (SD) practice and are frequently praised for their beneficial outcomes. Research has identified various benefits of ASD, including better customer requirements meeting (Vidgen and Wang, 2009), reduced time to market (Overhage and Schlauderer, 2012), and increased productivity (Tarhan and Yilmaz, 2014), which are summarized under the concept of *ASD business value* (Meckenstock et al., 2022). However, in addition to desirable consequences, research also suggests that ASD can have significant negative implications. Examples comprise increased stress for developers [S4, S64], technical debt [S12,S42], insufficient customer collaboration [S27,S67], and limited documentation [S10,S55]. Considering these negative effects reported in the literature, ASD methodologies also appear to have a dark side to them.

Although ASD research is a well-established field that has received significant attention since the introduction of the methodologies over two decades ago (Baham and Hirschheim, 2022; Dingsøyr et al., 2012), a distinct research gap seems apparent for this dark side of ASD compared to research on its business value. Thus far, progress has primarily been made towards systematizing the beneficial facets that constitute the business value of ASD (Meckenstock et al., 2022; Racheva et al., 2010). Moreover, advances towards explaining these benefits, i.e., *how* ASD creates benefits that jointly contribute to software quality (Alami and Krancher, 2022) or *how* pair programming stimulates team performance (Kude et al., 2019) have enriched our understanding.

In contrast, the literature lacks comprehensive systematizations and explanations for the occurrence of the negative aspects. *Four* shortcomings can be identified through neglect spotting (Sandberg and Alvesson, 2010) when evaluating related work on the topic. *First*, the only work that attempted to synthesize findings on a wide range of potential issues in ASD (Fitriani et al., 2016) remained superficial in the presentation of findings. This compilation also neglects key issues in ASD, i.e., lacking customer collaboration [S27] or stress for the developers [S37]. *Secondly*, only certain challenging areas of ASD have been thoroughly consolidated, including requirements engineering (RE) challenges (Inayat et al., 2015) or communication challenges in global

E-mail address: jan-niklas.meckenstock@uni-bamberg.de.

https://doi.org/10.1016/j.jss.2024.111966
Received 29 June 2023; Received in revised form 29 November 2023; Accepted 5 January 2024
Available online 26 January 2024

ASD (Alzoubi et al., 2016). However, as shown, ASD can negatively affect various aspects beyond requirements and communication, which are therefore not considered in these reviews and thus remain unclear. *Third*, findings that emerged with the growth in publications on ASD after 2016 (Baham and Hirschheim, 2022) are not included in these reviews. Emergent research streams on stress, resource depletion, and fatigue in ASD [S11,S47,S53] or negative effects on innovation [S3-S5, S35] are thus not represented, calling for an updated systematization. *Lastly*, an analysis of *how* different issues relate to each other to gain an understanding of the complex dark side of ASD generally remains to be developed yet.

As evidence on the downsides remains to be thoroughly consolidated and understood, it appears difficult to grasp *what* characterizes the dark side of ASD and *how* the associated issues arise. Gaining a better understanding of the negative aspects appears important, however, given the significant number of organizations that are yet immature regarding the application of ASD, as noted in the "Annual State of Agile" reports (VersionOne, 2018, 2019, 2020, 2021). Unexperienced practitioners unaware of these problems may thus be confronted with unexpected issues and their aggravating consequences. Additionally, scholars that aim at generating explanations for "*if, how, why and when*" (Baham and Hirschheim, 2022, p. 5) ASD methodologies deliver different outcomes also need to understand *what* the spectrum of negative outcomes comprises and *how* the complexity of the dark side is defined. Since research that addresses *what* facets constitute the business value of ASD and *how* certain benefits can be generated has recently made progress, an updated systematization of the negative consequences of ASD represents the next logical step. To add, the analysis of *how* these issues are related to one another seems necessary to close the described research gap and to enhance our understanding of ASD.

Hence, to systematize and better understand the issues that may occur in ASD, we conducted a qualitative systematic literature review (SLR) (Paré et al., 2015, 2023) following Kitchenham and Charters (2007) that incorporates findings from software engineering (SE) research, information systems (IS) and management science literature. The SLR presented in this paper investigates the following research questions:

- *What are the specific issues that characterize the dark side of ASD methodology application and what are the constituents in SD that are primarily affected by them?*
- *How do various issues occurring in ASD methodology use relate to each other and how do potential relationships define the complexity of the dark side of ASD?*

The SLR analyses 70 articles in a systematic analysis process that follows Gioia et al. (2013) and identifies 90 issues that are merged into 18 interrelated issue themes. The analysis allocates these issues to the respective constituent that seems predominantly affected by them, relying on an adapted model by Chow and Cao (2008). Furthermore, our work illustrates 65 relationships between these issues and elaborates on the complex effect mechanisms using a relationship diagram. Based on the systematization of the negative implications and the analysis of relationships between these issues, we develop five propositions that capture the quintessence of our findings on the dark side of ASD.

The contribution of our work is as follows: our analysis helps to systematize the breadth of issues described in the literature that may occur in ASD methodology use, addressing the question of *what* characterizes the downsides of ASD. Note that this systematization does not intend to discredit ASD methodologies for their potential shortcomings, but instead aims to provide more clarity on the spectrum of the possible negative outcomes, since the dark side has been less regarded in extant research. For academia, our work can thus aid future research in investigations on the concurrent positive and negative implications of ASD, as it complements findings on its business value (Meckenstock et al., 2022) with an up-to-date systematization of the opposite side of effects. We thereby also fulfill a prerequisite for investigations into *"if, how, why* and *when"* (Baham and Hirschheim, 2022, p. 5) ASD methodologies create different outcomes.

For practitioners, this systematization can draw their attention to the most pressing issues that they may have to act on in their respective cases. As such, especially the team, the process, and the product can be affected, considering the frequency of corresponding issues. In contrast to the primarily positive stance on ASD, particularly in the "Annual State of Agile" reports, our findings also explain these negative consequences, thereby preventing an overly positive perspective on ASD methodologies. Combined with the findings on the business value of ASD in Meckenstock et al. (2022), our work can thus provide a more balanced representation of the overall effects of the methodologies.

Finally, the identification of relationships among issues helps to better understand *how* the complex nature of the dark side of ASD arises. The analysis can therefore enable practitioners to identify root causes for certain issues to improve the application of ASD in their organization. The interplay among issues may meanwhile provide a new approach for scholars towards explaining *how* and *why* ASD methodologies either deliver the desired business value or cause problems.

The paper proceeds as follows: Section 2 illustrates the theoretical underpinnings of ASD, presents the analysis framework, and discusses related work. Section 3 contains the research approach behind our work. In Section 4, we present our findings on the issues in ASD and illustrate relations among them using a relationship diagram. Section 5 delineates the core contributions along five propositions, discusses implications for academia and practice, and addresses limitations. Section 6 concludes the paper.

2. Theoretical background

2.1. Agile software development methodologies

ASD methodologies such as Scrum, XP, Kanban, and scaled agile frameworks provide novel approaches for SD. Each methodology relies on different sets of practices as "its collective components" (Conboy, 2009, p. 338) that stimulate the realization of *agility* in the development process (Baham and Hirschheim, 2022). Extant research on ASD methodologies understands *agility* as the "ability to anticipate, create, learn from and respond to changes in user requirements through a process of continual readiness" (Baham and Hirschheim, 2022, p. 10). As a unified foundation, *agility* in ASD is based on the 12 principles captured by the Agile Manifesto (Beck et al., 2001). It is further specified by four core concepts that capture the essence of all ASD approaches regarding the anticipation and creation of and response to change as well as learning from these changes in the environment (Baham and Hirschheim, 2022).

The 12 guiding principles of the Agile Manifesto depict guidelines (Abrantes and Travassos, 2011; Baham and Hirschheim, 2022) on how to approach the development work in ASD. The four core concepts that highlight "incremental design and iterative development", together with "inspect and adapt cycles" whilst "working cooperatively / collaboratively / in close communication" and under "continuous customer involvement" (Baham and Hirschheim, 2022, p. 16) capture the characteristic essence of agility in SD. Both the guiding principles and the core concepts are operationalized through the application of methodology-specific practices and techniques (Baham and Hirschheim, 2022). As illustrated in Fig. 1, applying the practices generally results in various outcomes. Regarding this rather generic formulation of outcomes, research first illustrates that ASD methodologies can deliver *desirable* outcomes, i.e. by "contributing to perceived customer value (economy, quality, and simplicity)" (Conboy, 2009, p. 340). In addition, the various desirable outcomes can go beyond "time to market and software quality" (Baham and Hirschheim, 2022, p. 17). As such, research illustrates that ASD methodology use can result in, inter alia, increased job satisfaction for developers (Fitzgerald et al., 2006; Tripp et al., 2016), better customer requirements meeting (Fitzgerald et al.,

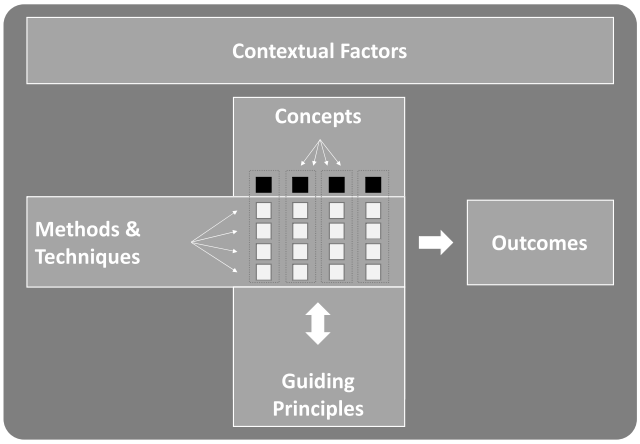


Fig. 1. Concept of Agility in Software Development (Baham and Hirschheim, 2022).

2006; Vidgen and Wang, 2009) and improved business-IT alignment (Elbanna and Murray, 2009; Overhage et al., 2011). As an aggregated form, scholars have determined that these various benefits constitute the *business value of ASD methodologies* (Meckenstock et al., 2022; Racheva et al., 2009, 2010). Research has further concluded that this business value entails "not only dollars" (Racheva et al., 2010, p. 131), but instead, the project team, the SD process, the customer, and the product outcomes all represent beneficiaries of the application of ASD methodologies (Meckenstock et al., 2022). ASD business value therefore depicts a multifaceted concept with various value dimensions and value recipients (Meckenstock et al., 2022).

Besides the ASD business value perspective, scholars have analogously identified *negative* outcomes that resulted from the application of the methodologies. Exemplary findings include technical debt [S12, S42], stress and pressure for developers [S11,S46,S64], documentation problems [S10,S61] and an overhead of unnecessary meetings [S53,S64, S65]. In contrast to the established business value perspective in ASD, however, the dark side of ASD appears somewhat neglected in extant research, as a comprehensive overview that addresses *what* negative implications can result is still lacking. Additionally, insights into *how* these issues arise, and which specific aspects of ASD methodologies may be causal for the negative implications, remain scattered in the literature. While research demands to investigate *"if, how, why* and *when* ASD impacts outcomes" (Baham and Hirschheim, 2022, p. 5), we argue that examining the question of *what* defines the spectrum of outcomes is a prerequisite for these investigations. Since the positive outcomes have already been integrated into a coherent concept (Meckenstock et al., 2022), unveiling the dark side of ASD represents the next logical step.

2.2. The dark side of agile software development methodologies

Besides illustrated problems such as technical debt [S12,S42], stress for developers [S11,S46,S64], insufficient documentation [S10,S61] and meeting overhead [S53,S64,S65], research has identified further potential issues that may occur in ASD methodology use. Frequent examples include problematic customer collaboration [S27,S67], productivity losses [S16], or delivery delays [S8,S28]. In addition, several constituents of ASD projects appear negatively affected by these issues, as the delivery delays stated in [S8,S28] imply negative consequences for the project outcomes, while the productivity losses [S16] seem to affect the development process. Further examples include the increased stress levels [S11,S46,S64] that especially concern the well-being of the development staff, whereas the frequently occurring issue (architecture) technical debt [S12,S42,S57] primarily finds reflection on the level of the product. Based on the issues that ASD methodology use may imply, a multidimensional spectrum of problems must be taken into consideration when addressing the dark side of ASD. In addition, the spectrum of manifestation levels, i.e., those constituents in SD that are potentially affected by these problems, appears similarly multifaceted.

To capture the multidimensionality of the dark side of ASD, a multifaceted analysis framework thus appears necessary. In this regard, we adapt the dimensions of a framework suggested by Chow and Cao (2008). The adapted model is illustrated in Fig. 2. While the framework by Chow and Cao (2008) originally investigates success factors that contribute to the success of ASD projects, their work simultaneously points out a collection of failure factors, i.e., issues, and dimensions where these occur. These issues appear to arise on similar *dimensions* as the success factors, which comprise the *organizational, people, process,* and *technical dimensions.* The dimensions also include several of the issues stated before, i.e., a lack of management support and inadequate corporate culture in the *organizational dimension* (Chow and Cao, 2008). Similarly, the lack of teamwork is embedded in the *people dimension* (Chow and Cao, 2008). Third, typical SD process-related issues such as problems caused by planning and progress-tracking mechanisms are listed in the *process dimension.* Lastly, the incorrect application of ASD practices and tools is assigned to *technical issues*, while contrastingly, the success factors include the appropriate application of typical ASD practices such as refactoring, testing, or coding standards (Chow and Cao, 2008). For our analysis, we thus adopt the following dimensions: *organizational issues, process issues,* and *technical issues.*

For the *people dimension*, we argue for a separation into a) *people* in the *development team* and b) the *customer* as a separate dimension. Since *customer-related* issues, i.e., "ill-defined customer role", "lack of customer presence" and "bad customer relationship" (p. 963) are somewhat ambiguously allocated to the *people* and *process dimension* in Chow and Cao (2008), we define a new dimension to delimit issues that primarily relate to the *customer*. As a result, the *people dimension* only addresses the *development team* and its *individual members*, calling for a relabeling of this dimension to *team and developer issues*. In addition to the *organizational, process,* and *technical issues* dimensions, we include a *customer issues* dimension that only addresses client-related problems and define a new dimension for the *team and developers*.

Lastly, the model hypothesizes that the success factors contribute to success in terms of *quality, scope, time,* and *cost*. For the context of our work, we invert the semantic logic of the model from a project success stance into a negative, project failure-oriented perspective. As such, we hypothesize that the presence of issues in the dimensions stated before can lead to project failure. Inverting the attributes of project success by Chow and Cao (2008) into corresponding failure attributes thus allows us to classify findings in the literature that suggest a non-performance of ASD projects. These may include poor software quality, schedule and budget overruns, as well as failing to meet requirements and other *outcome-related issues* in ASD projects. Concludingly, the analysis framework comprises the following dimensions: *team and developer issues, process issues, customer issues, technical issues, organizational context issues,* and the *project outcomes* that suggest a non-performance of ASD projects. The model and its levels are featured in Fig. 2.

2.3. Related work

In contrast to research on the business value of ASD, where recent studies have comprehensively systematized the spectrum of related beneficial aspects (Meckenstock et al., 2022), comparable works on the dark side of ASD appear somewhat lacking. In this regard, *four* distinct shortcomings can be distilled in the body of literature through neglect spotting (Sandberg and Alvesson, 2010) that call for a thorough, comprehensive, and up-to-date review of previous findings on the unwanted consequences of ASD.

First, the only study thus far with an unrestricted focus on the entire spectrum of negative implications of ASD remained on a high level of abstraction and rather coarse in the description of the findings. While the work by Fitriani et al. (2016) can be seen as an initial step towards

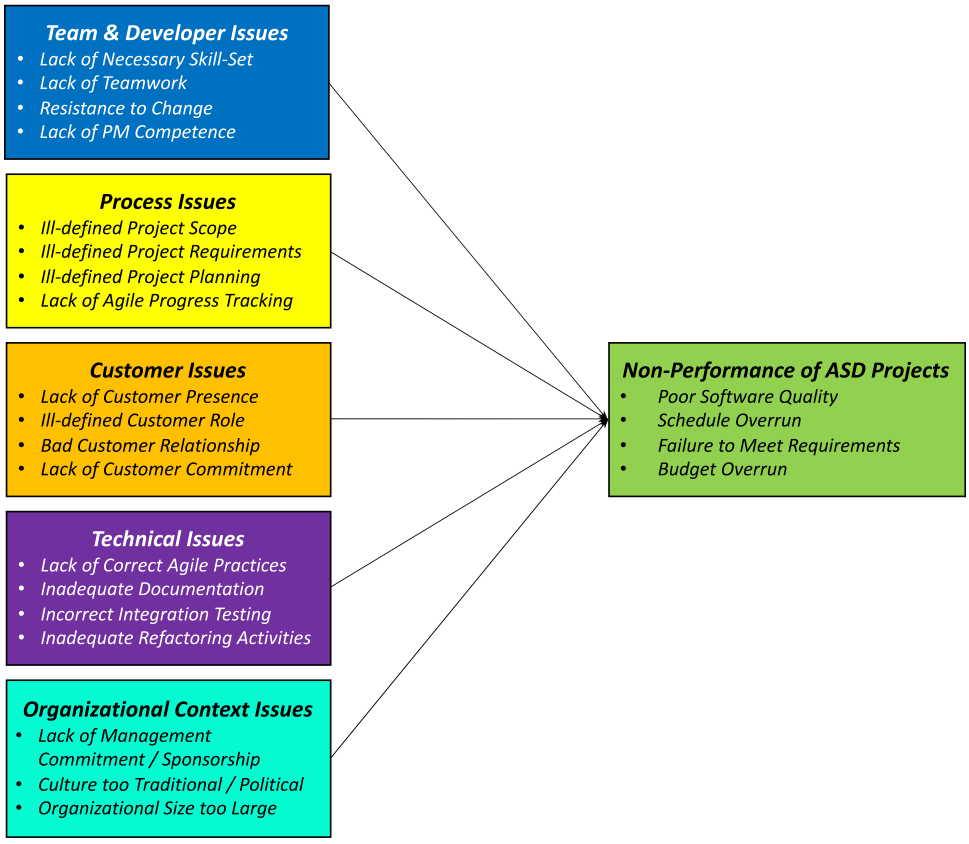


Fig. 2. Analysis Framework adapted from Chow and Cao (2008).

comprehensively examining the downsides of ASD by identifying 30 challenges, the analysis remains inexplicit and ambiguous in the delineation of these issues. Especially the generic denomination of challenges such as "process", "technical" and "communication" leaves room for interpretation as to what a certain challenge entails, since a variety of issues may be associated with it, which are however not clarified. Furthermore, it is unclear why the majority of challenges are considered "insignificant" (Fitriani et al., 2016, p. 160), solely based on their occurrence fewer than five times in a sample of only 20 selected papers. 75 % of the identified issues therefore remain unclear, since only 25 % are described in detail. In addition, the search string only covers a limited set of notions relatable to the dark side of ASD, thereby potentially omitting relevant findings. Lastly, this paper remains on a descriptive level and barely engages in explaining the occurrence of these issues. Consequently, a thorough understanding of the spectrum of issues in ASD was not established by this review.

As a *second* shortcoming, prior work that has conducted thorough reviews of issues in ASD only focused on very distinct challenging aspects. Examples include the review by Inayat et al. (2015) that systematized challenges particularly occurring in agile RE. Other areas beyond RE, however, are out of scope. The same applies to the studies by Jalali and Wohlin (2011) and Alzoubi et al. (2016), which focused on communication challenges in distributed, global ASD and SE, while not taking other problematic aspects into account. In addition, the investigated context of distributed SD represents a rather specific application scenario for ASD methodologies, thus not focusing on issues that generally occur with ASD. This is further corroborated by Hoda et al. (2017), since only three of the included studies exhibit a distinct focus on issues in ASD, while the remaining studies only address ASD-related issues, if at all, on a side note. Concludingly, while some distinct challenging aspects of ASD are very thoroughly assessed, a comprehensive systematization of the entire spectrum of occurring issues is still lacking in the literature.

As a *third* shortcoming that appears pressing in consideration of recent advances made in research on the negative aspects of ASD, all works discussed before only incorporated studies up to 2016. ASD research has seen a significant rise in publications in the SE-community since 2016 (Baham and Hirschheim, 2022), however, especially with new research threads having recently emerged. Examples include the negative implications of ASD on developer well-being, such as stress, resource depletion, or fatigue [S11,S47,S53]. A second stream that recently emerged pertains to the undesirable negative effects on innovation and organizational learning [S3-S5,S35]. These and other important aspects are therefore not included in the aforementioned studies, thus calling for an updated examination that also incorporates recently emerging research strands.

Lastly, an analysis that would help to explain *how* different issues occurring in ASD relate to each other is generally missing in the extant body of literature. To the best of our knowledge, no prior study has taken a comparable approach to explain *how* the different issues define the complexity of the dark side of ASD, which presents a further neglected aspect in research on the downsides of ASD methodologies.

Considering these shortcomings, the dark side of ASD methodologies altogether appears less regarded by extant research. This paper aims to address these deficiencies by providing an up-to-date systematization via an SLR that incorporates an unrestricted spectrum of potential issues arising in ASD. To add, this SLR significantly extends and overhauls previous reviews on the issues in ASD by incorporating recent findings that have advanced research on the focal subject. The SLR furthermore intends to thoroughly investigate the specificities of the identified issues and examine the relationships between them to address *what* characterizes the dark side of ASD and *how* its complex nature is defined. Understanding *how* these issues relate to each other and *how* they define the complexity of the dark side has generally not been addressed by extant research. Especially the analysis of these relationships and potential chain reactions among issues can hence provide significant

advancements for explanatory approaches on these complex effect-creation mechanisms in ASD methodologies, thus going beyond solely summarizing extant findings. Finally, by addressing the questions of *what* characterizes the dark side of ASD and *how* issues arise, a prerequisite for future investigations on *"if, how, why* and *when"* (Baham and Hirschheim, 2022, p. 5) ASD creates outcomes will also be fulfilled with this SLR, with a primary focus on the unwanted consequences.

3. Methodology

3.1. Overview of the systematic literature review

Since our research goal is to systematize the potential negative consequences of ASD, this SLR represents a systematic qualitative review (Paré et al., 2015, 2023) that integrates prior empirical qualitative or quantitative findings. The SLR relies on guidelines by Kitchenham and Charters (2007) to identify and systematize extant literature that discusses these negative aspects of ASD methodology use. Previous SLRs on ASD methodologies similarly applied these guidelines, i.e., Dybå and Dingsøyr (2008), Kupiainen et al. (2015), Selleri Silva et al. (2015) and Alzoubi et al. (2016). The qualitative content analysis is based on a grounded-theory-oriented approach (Gioia et al., 2013). The subsequent chapters elaborate on the literature search, selection of relevant articles, and the qualitative analysis of the selected contributions, as shown in Fig. 3 below. Fig. 4 meanwhile contains the complete search and selection process, from the initial number of potentially relevant results to the final literature sample for the analysis.

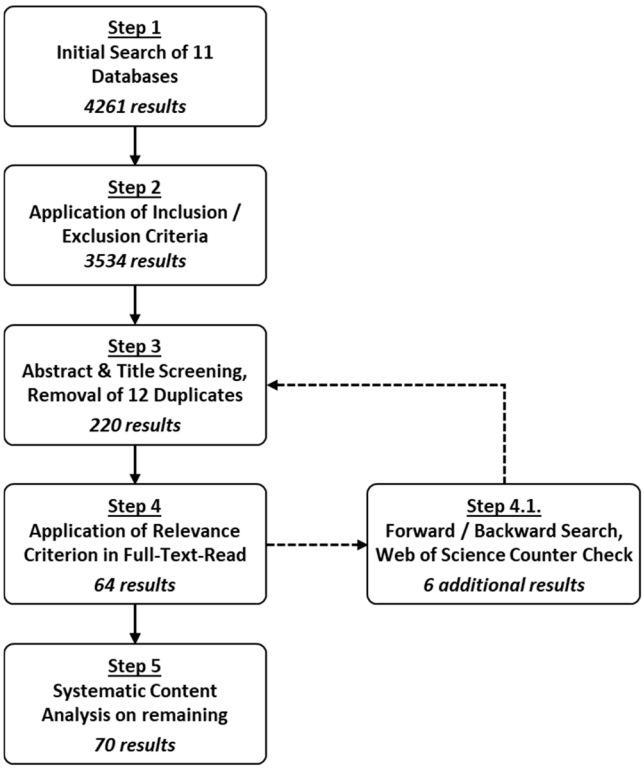


Fig. 4. Literature Search and Selection Process.

3.2. Systematic literature search and selection

To meet the requirement of systematic qualitative reviews for comprehensiveness (Paré et al., 2015), we considered an exhaustive selection of databases in the literature search. Our search included databases from SE and IS research, while also incorporating management science literature. We employed IEEE xplore, ACM digital library, ScienceDirect, and SpringerLink as distinct outlets for SE literature. The latter two outlets seem especially important to our research, as ScienceDirect contains several major SE journals, while SpringerLink features the annual AGILE and XP conferences that discuss important findings related to our topic. For IS research, the AIS electronic library and Taylor & Francis were selected, considering their focus on IS journals and conferences that have contributed to ASD research. Lastly, the EBSCO Business Source Ultimate and Wiley Online Library were selected due to their emphasis on management research. We furthermore consulted Google Scholar and the Web of Science in a subsequent snowballing procedure to identify additional relevant literature, as outlined later on.

To define a search string that identifies relevant contributions in the databases, we conducted an initial review of related literature on the dark side of ASD to determine relevant terms that relate to our research focus. From this screening, we identified "agile software development" and "agile development" as generic terms, "agile information systems development" as a more specific term used in IS literature, and "agile methods" as well as "agile practices" to define the targeted search environment. We observed that the term "agile" is also frequently used in several other research domains besides SD, i.e., agile supply chain management. To exclude those terms that would deliver results not related to our research questions, we extended the search string with an exclusion (**NOT**) operator. With our research focusing on the negative aspects in ASD, we then derived expressions that pertain to these problematic, unwanted consequences. We identified 22 different notions that embody the sematic meaning of *problem* or *issue*. Lastly, we added wildcards (*) to the search terms to maximize the coverage of the search string. The three main components were then merged into the search string below, while its individual elements are featured in the complete search string and in Table 1 on the next page.

(((*Generic ASD Terms*) **NOT** (*Excluded Terms*)) **AND** (*Issue-related Terms*))

The search string below was adapted to meet the requirements of the query builders of the databases, see the online appendix: https://figshare.com/s/20c1a6307d761c08e2a3.

(("agile software development" **OR** agile method* **OR** agile practi* **OR** "agile development" **OR** "agile information systems development") **NOT** ("agile manufacturing" **OR** "agile supply chain" **OR** "agile

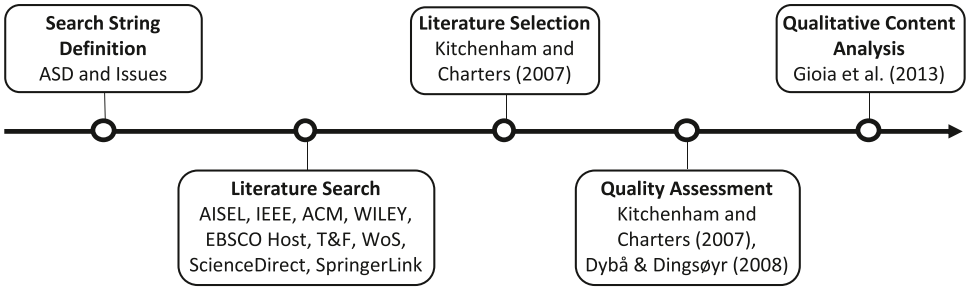


Fig. 3. Process for the Systematic Qualitative Review.

Table 1
Search string elements.

Generic ASD Term	Excluded Terms	Issue-related Terms
agile software development; agile method*; agile practi*; agile development; agile information systems development;	agile manufacturing; agile supply chain; agile engineering; organizational agility;	problem*; issue*; downside*; weakness*; difficulty; difficulties; drawback*; dark side*; concern*; dilemma*; disadvantag*; cons; complication*; deficiency; risk*; challeng*; stress*; negative*; reduc*; shortcoming*; obstacle; headache;

Table 2
Inclusion and exclusion criteria.

Criterion	Description of Inclusion Criteria (IC) / Exclusion Criteria (EC)
IC1	The language of the publication had to be English.
IC2	The publication had to be published by a journal or a conference.
IC3	The research item was published within the last 16 years (2022–2006) since ASD methodologies gained major attention in research from 2006 onwards (Baham and Hirschheim, 2022; Dingsøyr et al., 2012).
IC4	Only complete research articles were included; thus opinions, keynotes, viewpoints, and posters were excluded from the sample.
EC1	Studies that do not focus explicitly on problems that occurred in the application of ASD methodologies were excluded from the sample.
EC2	Studies that do not focus on ASD methodology problems were excluded.
EC3	Studies that focus on agility in general, not on ASD methodologies, were excluded.
EC4	Duplicates were excluded from the sample.

engineering" **OR** "organizational agility")) **AND** (problem* **OR** issue* **OR** downside* **OR** weakness* **OR** difficulty **OR** difficulties OR drawback* **OR** dark side* **OR** concern* OR dilemma* **OR** disadvantag* **OR** cons **OR** complication* **OR** deficiency **OR** risk* **OR** challeng* **OR** stress* **OR** negative* **OR** reduc* **OR** shortcoming* **OR** obstacle **OR** headache))

The literature search delivered 4261 initial hits. To define a final sample, we followed a systematic literature selection process suggested by Kitchenham and Charters (2007), featured in Fig. 4. We first derived inclusion and exclusion criteria from previous SLRs that followed the suggestions by Kitchenham and Charters (2007) to reduce the sample size initially. The criteria can be found in Table 2.

The application of these criteria resulted in 3534 potentially relevant articles. Next, we conducted a 2nd screening step and assessed both title and abstract to determine thematic relevance. In this step, we also eliminated duplicates, resulting in 220 articles for the full-text assessment. Of these articles, 156 articles were disregarded, as these studies did not sufficiently report on the negative aspects of ASD. From the initial sample, 64 articles fulfilled this relevance criterion in the 4th step. We performed an iterative forward and backward-search that follows the snowballing procedure by Wohlin (2014), using the citations and references of the selected articles. We relied on Google Scholar to identify additional literature relevant to our research that was cited in already selected publications, backed up with a citation search on the Web of Science. This snowballing procedure identified six additional articles, resulting in 70 articles that define the final sample. The complete database search and selection results are enclosed in Table 3.

In line with Paré et al. (2015) and Kitchenham and Charters (2007), we subsequently conducted a quality assessment. We relied on quality assessment criteria suggested by Dybå and Dingsøyr (2008), Kitchenham and Charters (2007), and others to evaluate the rigor, credibility, and relevance of the findings presented in the articles. The selected criteria are enclosed in Table 4 above. The final selection of articles appears to be of high quality, given that most articles met the quality criteria by Dybå and Dingsøyr (2008), as represented in Fig. 5. Following Hoda et al. (2017), we also defined a cut-off value of at least 50 % for the required quality appraisal points to allow inclusion in the review sample. The articles identified for this review exceeded this threshold without exception. The complete quality assessment results are enclosed in Table 5 in Section 4.2.

3.3. Qualitative literature analysis

The 70 articles were analyzed in a qualitative content analysis that

Table 3
Database search and selection results.

Database	Initial Search Results	Results after Hard Inclusion Criteria	Final Sample after Title / Abstract Screening & Content Analysis
AISEL	156	152	9
IEEE	2152	1959	9
ScienceDirect	464	228	17
EBSCO Host	592	350	8
Taylor & Francis	7	6	2
SpringerLink	537	504	15
ACM	332	318	2
WILEY	21	17	2
Snowballing Web of Science / Google Scholar			**6** results added to existing sample
Results Count	4261	3534	76 initial total Duplicates: 12 Snowballing: +6 Result: 70

Table 4
Quality assessment criteria.

Quality Aspect	Quality Criterion	Focus of Quality Criterion	Authors
Rigor	Q1	Theoretical Foundations	Kitchenham and Charters (2007)
	Q2	Data Collection, Analysis and Reporting	Rohunen et al. (2010), Usman et al. (2014)
	Q3	Limitations and Threats to Validity	Rohunen et al. (2010)
	Q4	Use of References	Dybå and Dingsøyr (2008)
Credibility	Q5	Aims of Research	Dybå and Dingsøyr (2008)
	Q6	Credibility and Statements of Findings	Kitchenham and Charters (2007)
	Q7	Common Thread in the Article	Kitchenham and Charters (2007)
Relevance	Q8	Contribution to Knowledge or Understanding	Kitchenham and Charters (2007)

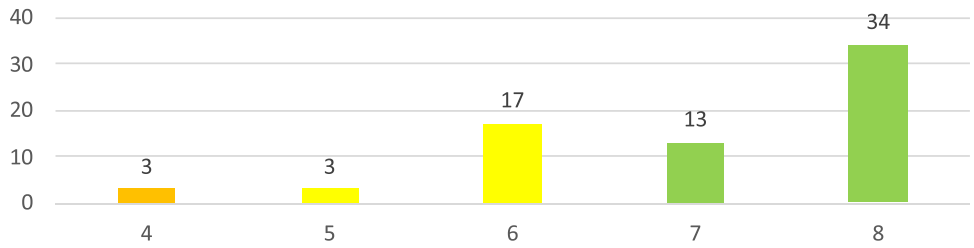


Fig. 5. Quality Assessment Scores.

followed three subsequent data extraction, consolidation, and mapping steps suggested by Gioia et al. (2013) to create a data structure that allows to better understand the negative side of ASD methodologies. The first step of the analysis (*open coding*) entailed the extraction of text fragments that contain statements on negative aspects that occurred in ASD. We extracted 595 text fragments in total, of which some indicated more than one potential downside. This *open coding* step examined the statements and initially identified specific keywords that described occurring issues, leading to an initial set of concepts. As the open coding procedure progressed, certain concepts were relabeled to further specify the underlying issue or merged to avoid doubling concepts that identify the same issue. Furthermore, some concepts were also divided into separate concepts to enable higher descriptive accuracy in the labeling of distinct issues that were previously grouped under the same notion, yet featured certain nuances that would require separation to ensure adequate representation of the data. We performed this step in three iterations to achieve a well-defined foundation for the subsequent coding steps. Upon completion of this iterative process, 90 1st order concepts that each represent a distinct issue of ASD methodologies were developed based on the extracted statements.

In the subsequent *axial coding* step, we aggregated related 1st order concepts to joint 2nd order issue themes. This step serves to identify relations between the previously defined 1st order concepts, based on semantic similarities. We first grouped 1st order concepts that share a similar characteristic regarding a superordinate negative aspect. As an illustrative example of a characteristic embodied by several 1st order concepts, *Inadequate Requirements Prioritization, Negligence of Non-Functional Requirements, Difficult Requirements Elicitation, and Inadequate Requirement Specification / Separation* were grouped together in a joint 2nd order theme labelled ***Requirements Engineering Problems***. All issues address somewhat related, yet slightly differentiated problems that specifically occurred in the RE process in ASD methodologies. Analogously, the 2nd order concept ***Physical & Emotional Health Problems*** was derived from the subordinate 1st order concepts *Stress, Developer Frustration, Demotivation of Developers, Exhaustion / Resource Depletion / Burnout* and *Reduced Well-Being*, since these issues all entail negative implications for the health of developers that apply ASD methodologies. The *axial coding* step was conducted iteratively to attain theoretical saturation (Strauss and Corbin, 1990; Wolfswinkel et al., 2017). We thoroughly reviewed the 1st order concepts and their classification to 2nd order themes, until no new concepts, themes and connections emerged. Upon reaching theoretical saturation, 90 1st order concepts defined in the previous step were merged into 18 2nd order value themes.

As a last step of the qualitative analysis, Gioia et al. (2013) suggest developing aggregate dimensions in a *selective coding* procedure to provide further levels of abstraction. In this step, we aggregated the 18 2nd order value themes according to the dimensions of the adapted model by Chow and Cao (2008), as described in Section 2.2. The framework helps to determine for which primary constituent involved in ASD projects the distinct issues manifest themselves. As an example, we aggregated the issue theme on ***Physical & Emotional Health Problems*** with other 2nd order themes such as ***Pressure Problems*** on the level of the development team, since these issues primarily affect the developers. 2nd order themes such as ***Product Quality*** and ***Architecture Problems*** were meanwhile grouped to the product level, since these themes and their associated 1st order concepts primarily imply negative consequences for the deliverables and the resulting project outcomes. Following this approach, we allocated the 18 2nd order issue themes to the six levels of our framework. Chapters 4.3 to 4.9 illustrate our findings along the 6 levels of the analysis framework and provide exemplary statements from the literature.

3.4. Analysis of relationships between identified issues

We lastly examined relationships between issues that occurred in ASD. To systematize these relationships, we focused on those statements that illustrate a certain issue, which in turn led to another problem, thus potentially depicting a causal relationship. As an illustrative example, [S5] described that the "constant pressure to deliver and the presence of short feedback loops within each work iteration increased the time pressure and the debilitating effects of stress that accompanied it" (p. 68). In this case, a relationship between the issues *Iterative Delivery Pressure for Developers, Tight Deadlines / Time Pressure* and *Stress* seems identifiable. Similarly, [S23] observed that "volatile and late requirements introduce uncertainty in effort estimates, [which] are perceived to be a direct cause of delays" (p. 6). In this statement, the issues *Volatile Requirements, Inadequate Effort Estimation* and *Delivery Delays* appear to be connected. In total, we found 256 text fragments that seem to illustrate relationships among issues. Based on the identified relationships, we developed a diagram to visualize the complex interplay of the issues that may occur in ASD. Due to the considerable number of identified relationships, the diagram presented in Section 4.10 only contains the most frequently identified connections among the different issues that appeared of significant importance. To be included, a certain relationship needed to be illustrated in at least two separate publications. Doing so allows us to show how the most pressing issues in ASD relate to one another, while not overwhelming the reader due to the number of identified relationships. The online appendix contains additional illustrative quotes from the sample to describe these relationships.

4. Results

We describe the review sample along quantitative aspects, followed by the results of the quality assessment. Subsequently, we illustrate the findings of our analysis in Section 4.3 using the adapted framework by Chow and Cao (2008). Sections 4.4 to 4.9 contain detailed descriptions of the issues reported in the literature. Lastly, we describe identified relationships between issues and provide a relationship diagram (Section 4.10).

4.1. Quantitative overview of the sample

We provide an overview of the publication venue, year of publication, applied ASD methodology, research approach, level of detail of context descriptions, industry or business domain where the studies were conducted in, and the ranking in the JOURQUAL 3 and ABDC ranking. All quantitative aspects are also included in the online appendix.

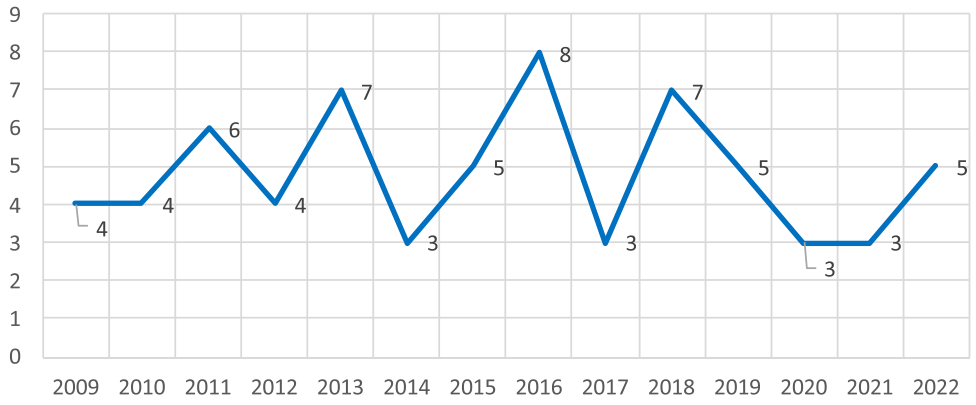


Fig. 6. Number of Published Papers per Year.

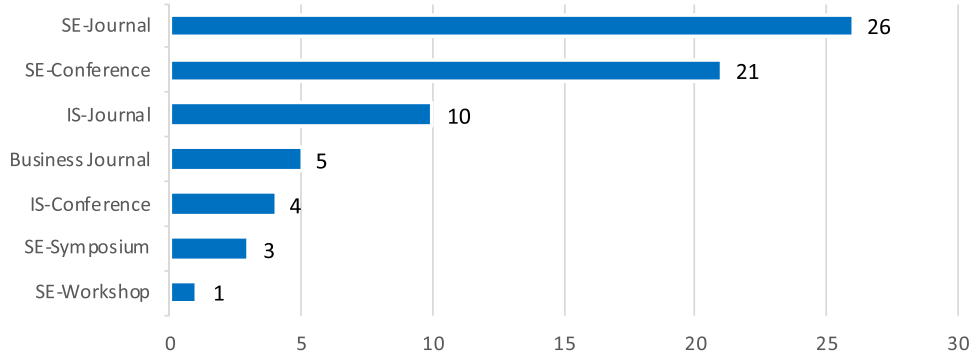


Fig. 7. Publication Outlets.

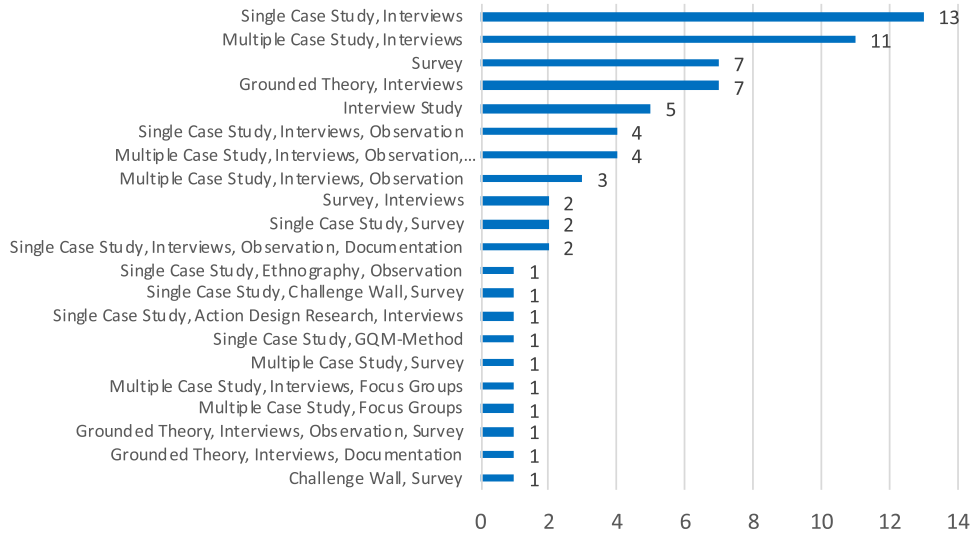


Fig. 8. Applied Research Methodologies.

Most articles were published between 2013 and 2018 (33/70, 47 %) and stem from the SE-domain (73 %), with 26 SE-journals (37 %) and 20 papers from SE-conferences (29 %), see Figs. 6 and 7. Only 14 studies (20 %) were published by the IS-community, accentuating the predominant stance of the SE-community in ASD research (Baham and Hirschheim, 2022). We evaluated the sample against two rankings. 42 (60 %) of the 70 articles are at least included in one of the rankings, while the other 28 articles are not ranked. This is not surprising, however, as they mostly stem from smaller conferences.

Most studies applied a qualitative research approach (59/70 articles), while 10 articles featured a quantitative approach. One article followed a mixed form of quantitative and qualitative research. Most articles based on qualitative research employed a single (13) or multiple case study (11), followed by questionnaire-based surveys (8) in quantitative research and 7 articles using grounded theory. Overall, case studies or grounded theory appear to be the most utilized approach, see Fig. 8. For quantitative approaches, surveys are more prominent (8), while confirmatory research strategies appear particularly rare, as only

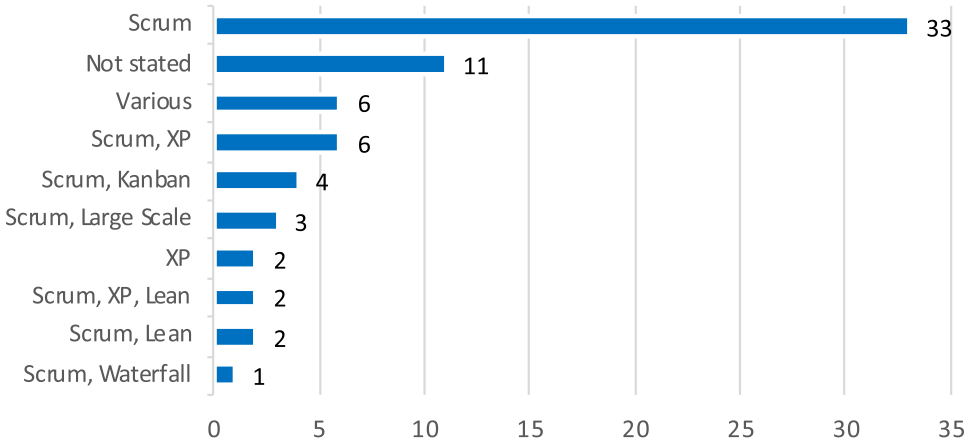


Fig. 9. Applied ASD Methodologies.

3 studies employed an explanatory approach, such as PLS-SEM [S53], multi-level regression [S35], or multi-level path analysis [S11].

Regarding the applied ASD methodologies, Scrum was applied in 33 of 70 cases, while a combination of Scrum with other methodologies such as XP, Kanban, and Lean was found in 18 other publications. The remaining 19 studies either applied XP, a combination of various ASD practices, or only stated that ASD methodologies were applied, whilst not reporting what exact method was employed. Fig. 9 contains a detailed summarization of the applied ASD methodologies.

We furthermore inspected the level of detail in the description of contextual conditions in the studies, as reported in the online appendix. Only 15 of 70 studies provided an in-depth description of the context, while almost 50 % elaborated on a rather low level of detail. In addition, the studies rarely reflected upon the role of these contextual factors in regard to their effect on the occurring issues. Lastly, we assessed the business domains or industries in which the studies were conducted, as stated in the contextual case description of the examined articles. Accordingly, 18 studies were conducted in the SD industry in the sense that the organizations operated as software providers for various clients, followed by 12 cases from the telecommunications industry. Cases that focused on organizations from multiple industries or business domains, especially in multiple case study scenarios, were grouped as "various industries" and constitute a third of the investigated studies (25/70 studies). 11 further studies stem from other industries, i.e., automotive, financial services or consultancy. 6 cases did not state in what industry their research was conducted. Fig. 10 presents the distribution of the different industries that were reported in the literature.

4.2. Quality assessment results

This section presents the results of the quality assessment. The following page summarizes the quality assessment results in a tabular format and provides insights into the scores awarded for each article along the 8 criteria defined for the quality appraisal. As can be inferred from Table 5 on the following page and Fig. 5 in the methodology Section 3.2, all assessed articles were evaluated with at least 4 points. The defined cut-off value of at least 4 points, adopted from Hoda et al. (2017), was therefore surpassed by all scrutinized articles. To add, the average score for the quality of assessed articles equals 7 points, indicating a high quality of selected contributions. Therefore, the complete final sample was admitted to the qualitative content analysis.

4.3. Overview of identified issues

In this section, we provide a graphical visualization that contains all identified issues that can potentially arise in ASD methodologies. Fig. 11 illustrates the spectrum of potential problems as the first result of our work. The figure relies on the adapted model by Chow and Cao (2008) and depicts six levels of issue manifestation, which contain 18 2nd order themes that are based on the 90 identified 1st order concepts. The subsequent chapters illustrate all issues featured in Fig. 11 together with a corresponding table (Tables 5–9) that contains the studies reporting on a specific issue and the frequency of occurrence in the literature. The tables employ the ID assigned to the studies during the quality assessment, as represented in Table 5. Illustrative quotes from the literature sample in the subsequent chapters follow this approach.

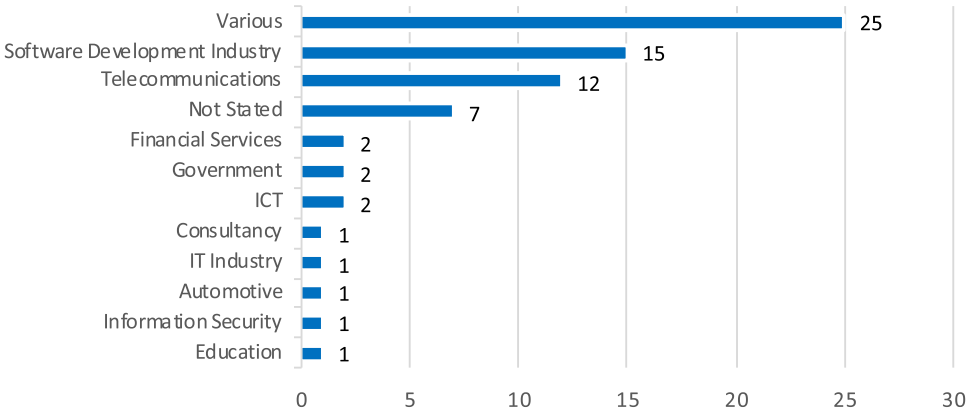


Fig. 10. Business Sectors and Industries.

Table 5
Quality assessment results.

ID	Authors	Year	Quality Score	Q1 Theoretical Foundations	Q2 Data Collection, Analysis & Reporting	Q3 Limitations Threats to Validity	Q4 Use of References	Q5 Aims of Research stated	Q6 Credibility of Findings	Q7 Common Thread in Article	Q8 Contribution to Knowledge
1	Alahyari et al.	2018	**8**	1	1	1	1	1	1	1	1
2	Alsaqaf et al.	2018	**6**	0	1	1	1	1	1	0	1
3	Annosi et al.	2022	**8**	1	1	1	1	1	1	1	1
4	Annosi et al.	2016	**8**	1	1	1	1	1	1	1	1
5	Annosi et al.	2020	**4**	1	0	1	0	0	1	0	1
6	Asnawi et al.	2011	**6**	1	1	0	1	1	1	0	1
7	Babar	2009	**7**	1	1	1	1	1	1	0	1
8	Badampudi et al.	2013	**6**	0	1	1	1	1	1	0	1
9	Basten et al.	2016	**8**	1	1	1	1	1	1	1	1
10	Behutiye et al.	2021	**8**	1	1	1	1	1	1	1	1
11	Benlian	2021	**7**	1	1	1	1	0	1	1	1
12	Codabux & Williams	2013	**6**	1	1	0	0	1	1	1	1
13	Conboy et al.	2011	**4**	0	1	0	0	1	1	0	1
14	Coyle et al.	2015	**7**	1	1	0	1	1	1	1	1
15	Dasanayake et al.	2019	**8**	1	1	1	1	1	1	1	1
16	de O. Melo et al.	2013	**8**	1	1	1	1	1	1	1	1
17	Drury et al.	2012	**7**	1	1	0	1	1	1	1	1
18	Elbanna	2014	**6**	1	1	0	1	1	1	0	1
19	Flouri & Berger	2010	**6**	1	1	1	1	0	1	0	1
20	Ghobadi & Mathiassen	2016	**8**	1	1	1	1	1	1	1	1
21	Gregory et al.	2016	**7**	0	1	1	1	1	1	1	1
22	Gregory et al.	2015	**6**	0	1	1	1	0	1	1	1
23	Hannay & Benestad	2010	**6**	0	1	1	1	1	1	0	1
24	Heikkilä et al.	2015	**8**	1	1	1	1	1	1	1	1
25	Heikkilä et al.	2017	**8**	1	1	1	1	1	1	1	1
26	Hess et al.	2018	**6**	1	1	0	1	1	1	0	1
27	Hoda et al.	2011	**7**	1	1	1	1	0	1	1	1
28	Hoda et al.	2016	**8**	1	1	1	1	1	1	1	1
29	Hoda et al.	2012	**8**	1	1	1	1	1	1	1	1
30	Holvitie et al.	2018	**8**	1	1	1	1	1	1	1	1
31	Kalenda et al.	2018	**8**	1	1	1	1	1	1	1	1
32	Kalinowski et al.	2016	**4**	0	1	0	1	0	1	0	1
33	Kamei et al.	2017	**6**	0	1	1	1	1	1	0	1
34	Katumba & Knauss	2014	**7**	1	1	0	1	1	1	1	1
35	Khanagha et al.	2022	**8**	1	1	1	1	1	1	1	1
36	Kopczynska et al.	2022	**7**	1	1	0	1	1	1	1	1
37	Laanti	2013	**8**	1	1	1	1	1	1	1	1
38	Laanti et al.	2011	**8**	1	1	1	1	1	1	1	1
39	Lagerberg et al.	2013	**5**	0	1	1	1	0	1	0	1
40	Mangalaraj et al.	2009	**7**	1	1	0	1	1	1	1	1
41	Marchenko & Abrahamsson	2008	**8**	1	1	1	1	1	1	1	1
42	Martini et al.	2015	**8**	1	1	1	1	1	1	1	1
43	Matthies & Dobrigkeit	2020	**6**	1	1	0	1	1	1	0	1
44	Matthies et al.	2019	**6**	1	1	0	1	1	1	0	1
45	McAvoy & Butler	2009	**7**	1	1	0	1	1	1	1	1
46	McHugh et al.	2011	**8**	1	1	1	1	1	1	1	1
47	Meier et al.	2018	**5**	0	1	1	0	1	0	1	1
48	Moe & Aurum	2008	**6**	1	1	0	1	1	1	0	1
49	Moe et al.	2019	**7**	1	1	0	1	1	1	1	1
50	Moe et al.	2010	**8**	1	1	1	1	1	1	1	1
51	Moe et al.	2012	**8**	1	1	1	1	1	1	1	1
52	Mohallel & Bass	2019	**5**	0	1	0	1	1	1	0	1
53	Mueller & Benlian	2022	**8**	1	1	1	1	1	1	1	1
54	Nuottila et al.	2014	**8**	1	1	1	1	1	1	1	1
55	O'Donnell & Richardson	2008	**6**	1	1	0	1	1	1	0	1
56	Olszewska et al.	2016	**8**	1	1	1	1	1	1	1	1
57	Paasivara et al.	2018	**8**	1	1	1	1	1	1	1	1
58	Petersen & Wohlin	2009	**8**	1	1	1	1	1	1	1	1
59	Rahy & Bass	2020	**7**	0	1	1	1	1	1	1	1
60	Ralph & Shportun	2013	**8**	1	1	1	1	1	1	1	1
61	Ramesh et al.	2010	**6**	0	1	0	1	1	1	1	1
62	Rodriguez et al.	2012	**8**	1	1	1	1	1	1	1	1
63	Schön et al.	2017	**6**	0	1	1	1	1	1	0	1
64	Singh & Strobel	2022	**8**	1	1	1	1	1	1	1	1
65	Stray et al.	2016	**8**	1	1	1	1	1	1	1	1
66	Stray et al.	2011	**7**	1	1	0	1	1	1	1	1
67	Van Waardenburg & van Vliet	2013	**8**	1	1	1	1	1	1	1	1
68	Vanhala & Kasurinen	2019	**8**	1	1	1	1	1	1	1	1
69	Waterman et al.	2015	**6**	0	1	1	1	0	1	1	1
70	Wiesche	2021	**8**	1	1	1	1	1	1	1	1
	TOTAL:			**55**	**69**	**50**	**66**	**62**	**69**	**51**	**70**

4.4. Team and developer issues

Issues concerning the team and developers include ***Collaboration Problems, Physical and Emotional Health Problems, Pressure Problems, Competence and Resource Problems, Team Cohesion,*** and ***Team Organization Problems***.

First, we address ***Collaboration Problems*** along *Inefficient Communication, Reduced Quality of Shared Information, Reduced Interaction among Team Members,* and *Lengthy Onboarding of new Members*.

Multiple studies reported *inefficient* or *difficult communication* [S26, S60]. Distributed settings appeared especially susceptible to communication issues [S15] due to "language barriers and cultural differences" [S15, p. 9]. Communication breakdowns can also be caused by "turnover of personnel, rapid changes to requirements, non-availability of appropriate customer representatives and growing complexity of the application" [S61, p. 467].

Reduced interaction among team members is a related issue. Affected by the "tight sprints schedule with little time for interaction" [S20, p. 14], [S59] found that "team members focus on completing their own user stories and tend to neglect requests from members of other teams" (p. 33). Distribution also hinders interaction between developers [S15, S54], as "the cooperation was not on the same level as the co-located teams" [S54, p. 77].

We also found evidence of a *reduced quality of shared information*. The pressure to report daily seems to be one cause [S64], as well as time pressure [S14] and communication with multiple handovers [S15,S59] impacting information quality.

Additionally, *onboarding of new members* can be challenging. While current members "know each other's strengths and weaknesses and [their] roles within the team" [S19, p. 15], new members need time to "get into the rhythm of the team" [S16, p. 418]. Multiple studies reported a loss of productivity as a direct consequence [S16,S18,S19].

For the theme on ***Physical and Emotional Health Problems***, we discovered *Stress, Reduced Well-Being, Exhaustion / Resource Depletion / Burnout, Developer Frustration* and *Demotivation of Developers.*

Developers generally appear to perceive ASD as more *stressful* [S4, S52]. Reporting progress in meetings [S64,S65] and delivering working code daily were observed as stressors [S5]. Also, the "increase in visibility and accountability" [S46, p. 102] causes individuals to stress themselves "to deliver on what was promised" [S46, p. 102]. Overall, ASD rarely seems to allow developers to "rest or take a break" [S64, p. 14].

As a consequence, *reduced well-being* among developers may result [S11,S65,S68]. Increased stress appears to "[deplete developer] resources and makes them feel 'burned out', ultimately impairing their developer well-being" [S11, p. 574]. In addition, multiple daily meetings can result in fatigue [S65], while the workload in short iterations seems to have similarly exhausting effects [S68].

Furthermore, we found *frustration* and *demotivation of developers*. Developers seem to experience frustration due to unavailable customers [S27,S46], pressure to report progress [S46], distorted information, negative client feedback [S59], and having to conduct refactoring before coding [S53]. Inconsistent workloads [S55] and fragmented tasks resulting in limited progress [S46] were also perceived as frustrating. Similarly, lengthy meetings demotivate the team members [S46]. Overall, adverse emotional effects seem to be particularly provoked by meetings and the pressure to report during such events [S46,S53,S64].

Next, we discuss ***Pressure Problems*** that occur within the SD team along *Iterative Delivery Pressure for Developers, Increased / Unbalanced Workloads, Reporting Pressure, Exposure of Developer Shortcomings,* and *Peer Pressure.*

The most frequently cited problem pertains to *iterative delivery pressure*. Teams experience "constant pressure to deliver and the presence of short feedback loops within each work iteration [increases] the time pressure" [S5, p. 68]. Furthermore, "short iteration lengths or an extremely high and unsustainable development velocity […] can cause excessive iteration pressure" [S29, p. 628]. Developers may therefore neglect tasks such as refactoring [S51], code documentation [S52], task progress documentation [S64] and learning as well as innovation [S4], thus also reducing the product quality [S25,S52].

Developers may furthermore experience *increased / unbalanced workloads*, as they "work overtime when the pressure is very intense" [S46, p. 98] and thus exceed the 40-hour week [S55]. As a result, the product quality declined [S55], developers suffered from exhaustion [S65,S68], and improvement opportunities were neglected [S53]. Together, delivery pressure [S46,S53], task fragmentation and context switching [S8,S34], and volatile requirements [S55] may "[result] in the workload of individual developers rapidly increasing" [S55, p. 20].

According to [S46, p. 98], "agile practices require team members to report daily on their progress", implying *reporting pressure*. For developers, it was reported that "it stresses [them] out having to attend a meeting every day to report [their] progress" [S52, p. 361]. Additionally, reporting minimal progress due to reasons beyond their control, such as customer unavailability [S46], can lead to frustration.

A related concern is the *exposure of developer shortcomings*. According to [S13], "stand-up meetings, onsite customers, and the use of storyboards and whiteboards made developer shortcomings visible to the rest of the team" (p. 49). Demonstrating their work while "not many people could recognize any improvements" [S41, p. 22], may especially cause developers with low self-esteem [S13] to feel uncomfortable [S65].

Lastly, there is evidence of *peer pressure* within ASD teams. As such, "individuals felt a certain amount of peer pressure at daily stand-ups and retrospectives to be seen to be making progress" [S46, p. 98]. [S4] noted that feedback loops "[generate] stress and pressure among team members" (p. 529). A result may be reduced innovativeness of the team, as "peer pressure is not beneficial for innovative team output" [S35 p. 9].

Subsequently, we describe ***Competence and Resource Problems***, comprising *Loss of Knowledge / Drain of Competence, Loss of Resources, Loss of Cross-Functionality of Developers,* the *Jack-of-All-Trades Problem, Limited Dissemination of Functional Knowledge,* a *Negligence of Learning / Improvement Opportunities,* and *Reduced Innovativeness and Creativity*.

First, essential *knowledge* of the developers can be *lost* in case of staff fluctuation [S6,S15,S54]. Compensating for this *loss of knowledge* appears difficult, as new members "have to get up to speed and to get to know the system" [S19, p. 15].

This *loss of resources* may provide further challenges. When personnel depart a project, i.e., due to staff fluctuation [S18], the estimated development time may be impacted [S8]. Multi-project environments, in particular, are prone to *resource loss* [S51].

Two further issues concern the *loss of cross-functionality of developers* and the *jack-of-all-trades problem*. The former is likely a result of inadequate self-assignment of tasks, as developers "selected simple and familiar tasks, [which] would lead to increased specialization and a loss of cross-functionality in the team" [S28, p. 14]. Inadequate task assignment "made [developers] even more specialized" [S65 p. 154], causing collaboration issues [S41] and a lack of redundancy [S65]. In contrast, the *jack-of-all-trades problem* implies cross-functionality without deepened levels of expertise. Developers in ASD are expected to be a "master of all trades" [S13, p. 51], yet several studies demonstrate that ASD causes developers to be "at best mediocre in doing a lot of things" [S3, p. 17].

Limited dissemination of functional knowledge seems related to the demand of handling any task. Developers face the need to possess knowledge about various aspects of development, which is not always present [S4]. As a potential consequence, [S23] reported that a "lack of such knowledge impedes well-designed solutions, [which] resulted in erroneous code" (p. 7).

Several studies illustrate the *negligence of learning* and *improvement opportunities* due to delivery and time pressures [S4,S5]. As such, "excessive pressure resulted in a neglect of learning and improvement" [S29, p. 631]. Additionally, "the lack of time for continuous learning" [S34, p. 9] may lead to the negligence of opportunities to enhance the process [S53].

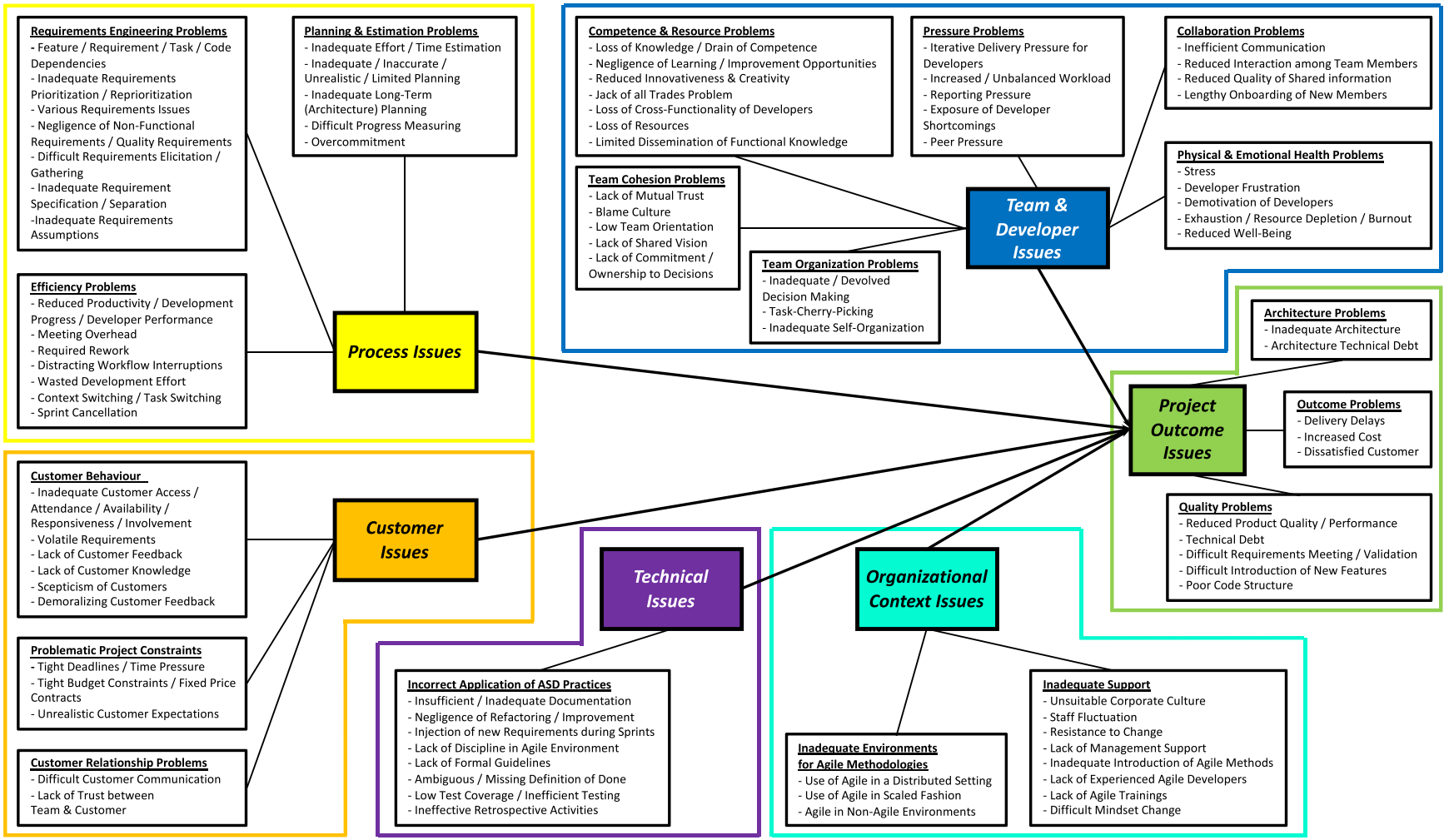


Fig. 11. Identified Issues in the Qualitative Content Analysis.

Table 6
Team and developer issues.

Issue Theme	Issue	Studies in our Sample	∑
Collaboration Problems	Inefficient Communication	[10,13,15,20,21,23,26,49,60,61,67,68]	**12**
	Reduced Interaction among Team Members	[15,20,41,49,50,51,54,59]	**8**
	Reduced Quality of Shared Information	[1,14,15,59,64]	**5**
	Lengthy Onboarding of New Members	[16,18,19]	**3**
Physical & Emotional Health Problems	Stress	[4,5,11,37,38,39,46,47,52,64,65,70]	**12**
	Developer Frustration	[27,45,46,51,53,55,59,70]	**8**
	Demotivation of Developers	[3,44,46,55,59,64]	**6**
	Exhaustion / Resource Depletion / Burnout	[11,65,68]	**3**
	Reduced Well-Being	[11]	**1**
Pressure Problems	Iterative Delivery Pressure for Developers	[2,3,4,5,8,11,14,23,25,27,29,39,46,51,52,55,59,64,66]	**19**
	Increased / Unbalanced Workload	[8,34,37,38,46,53,55,59,65,68]	**10**
	Reporting Pressure	[1,14,46,50,52,64,65]	**7**
	Exposure of Developer Shortcomings	[13,41,46,59,65]	**5**
	Peer Pressure	[4,35,46]	**3**
Competence & Resource Problems	Loss of Knowledge / Drain of Competence	[3,5,6,8,15,19,51,54]	**8**
	Negligence of Learning / Improvement Opportunities	[4,5,29,34,53,66]	**6**
	Reduced Innovativeness & Creativity	[3,4,5,35,64]	**5**
	Jack of all Trades Problem	[3,4,13,34]	**4**
	Loss of Cross-Functionality of Developers	[3,28,41,66]	**4**
	Loss of Resources	[8,51]	**2**
	Limited Dissemination of Functional Knowledge	[4,23]	**2**
Team Cohesion Problems	Lack of Mutual Trust	[1,15,21,48,50,57]	**6**
	Blame Culture	[13,44,45,55,64,65]	**6**
	Low Team Orientation	[48,50,51,57,66]	**5**
	Lack of Shared Vision	[23,49,50]	**3**
	Lack of Commitment / Ownership to Decisions	[17]	**1**
Team Organization Problems	Inadequate Decision Making	[1,13,17,20,51]	**5**
	Task-Cherry-Picking	[13,28,66]	**3**
	Inadequate Self-Organization	[13,28,47]	**3**

Similarly, time pressure seems to impede *innovation efforts* [S3-S5]. When teams are "focused on delivering and meeting deadlines, [it] is linked to lower innovative team output" [S35, p. 9]. As a result, the decreased ability for organizational innovation can result in fewer "new patents and product improvement ideas" [S5, p. 66].

We found several problems related to ***Team Cohesion*** along with a *Lack of Mutual Trust,* the existence of a *Blame Culture, Low Team Orientation,* a *Lack of Shared Vision,* and a *Lack of Commitment / Ownership to Decisions.*

First, a *lack of mutual trust* may result from the absence of co-location [S21,S57]. This may particularly occur in distributed settings [S15] and under permanent monitoring [S1,S50]. Working in distribution appears particularly problematic, as "people you were supposed to collaborate with changed constantly […], hindering the development of trust and slowing down team building" [S57, p. 16].

Additionally, a *blame culture* points towards weakened team cohesion. Blaming seems prevalent in both daily meetings and retrospectives [S13,S64,S65], where "a team is pointing fingers and assigning blame to team members instead of constructively searching for solutions" [S44, p. 5]. Also, blaming may occur in SD [S55] due to bugs in the code, thereby "complicating relations between individuals" [S55, p. 20].

Low team orientation may occur when developers "[prioritize] individual goals over team goals" [S50, p. 468]. Team members reported a lack of team spirit [S57] as developers tend to focus on their own agenda, leading to reductions in mutual trust [S48]. *Low team orientation* was also apparent when individuals participated in meetings but did not pay attention to other participants [S51,S66].

ASD Teams may furthermore experience a *lack of shared vision*, which can affect their delivery efficiency [S23]. This often entails the absence of a shared goal, which may indicate a lack of alignment, especially in larger development scenarios [S49].

As individuals concentrate on their personal agendas, [S17] noted a *lack of commitment and ownership to decisions* by the team. In this regard, "participants reported cases where the team makes a decision, but nobody really took ownership of seeing it through" [S17, p. 1247]. Consequently, the "implementation of those decisions left a lot to be desired" [S17, p. 1247].

Finally, we address issues on ***Team Organization***, specifically *Inadequate Decision Making*, the *Abilene Paradox / Group Think Problem, Inadequate Self-Organization, and Task-Cherry-Picking*.

Unavailable information appears to be the primary cause for *inadequate decision-making* [S1,S17,S51]. Additionally, decisions made "in development without consulting [the] client due to tight sprint schedules" [S20, p. 15] may be flawed and result in further issues in the implementation.

As a specific issue, [S45] illustrates the *Abilene paradox* in ASD projects and the *problem of groupthink*. Related to the *Abilene paradox*, decision-making may be thwarted, as "a collective decision is made that is actually contrary to the views of all members" [S45, p. 380]. Regarding *groupthink*, on "each occasion team unity and cohesion won out over good decision making" [S45, p. 377]. Both issues indicate that decision-making in ASD can sometimes be complicated.

Self-organization also appears to be problematic [S13] and relates to the *task-cherry-picking-problem*, as "people were picking tasks they shouldn't have" [S13, p. 54]. [S66] similarly illustrates that "that some tasks did not get picked, even when it had the highest priority" (p. 154). As potential reasons for this behavior, "specialization […], comfort and prior experience" [S28, p. 15] were identified. The consequences can include "increased specialization and a loss of cross-functionality" [S28, p. 16], and stress [S47].

4.5. Development process issues

This section illustrates issues in the SD process, including ***Efficiency***

Problems, Planning and Estimation Problems, and ***Requirements Engineering Problems***.

For ***Efficiency Problems***, we identified *Reduced Productivity / Development Progress / Developer Performance, Meeting Overhead, Distracting Workflow Interruptions, Task / Context Switching, Required Rework, Wasted Development Effort* and *Sprint Cancellations.*

The most prominent issue concerns *reduced productivity and developer performance*. Productivity seems particularly affected by team member turnover [S16], unavailability of customers and their requirements [S27,S67], sprint cancellation [S28], task dependencies [S28,S59] and interruptions due to meetings [S46,S50,S53,S70]. Therefore, reduced productivity may also affect timely product deliveries [S28,S46,S67].

Meeting overhead is a similarly common issue. While meetings are key for information exchange, exceeding their time limit [S46,S64] or too many meetings [S53,S65] can lead to significant overhead [S46] and may be perceived as disruptive [S46,S53,S70]. Distributed settings are especially prone to problems, since time differences can result in inconvenient meeting times for participants [S64,S65]. Many participants thus questioned the value of frequent meetings [S46,S53], perceiving them as "a waste of time" [S48, p. 80].

Other aspects affecting efficiency include *distracting workflow interruptions*, i.e., from the work environment, on-site customers, coding-related obstacles [S70], and collaboration-oriented practices [S53]. Interruptions during coding seem especially detrimental, as "[programming] requires deep concentration, [thus] the recovery time was longer" [S65, p. 29]. Developers generally seem to struggle to complete their work due to frequent interruptions [S46,S53,S64].

Additionally, *task / context switching* "is perceived to further decrease the productivity" [S23, p. 8]. Task switching frequently occurs when "people are participating in many projects at the same time and often need to jump from one task to the next" [S34, p. 9], causing a loss of productivity [S23,S64].

Furthermore, *required rework* resulting from technical debt, poses a significant efficiency-related challenge [S10,S12,S61]. To add, when the customer is not available and the team has to make assumptions, not meeting the actual requirements results in additional rework [S18,S27, S67].

Wasted development effort implies that completed work does not culminate in the final product [S1]. Effort may be wasted when teams are not aligned and redundant work is done [S23]. In a dedicated study on waste in ASD, [S1] found a variety of waste forms, including creating a wrong product, overly complex solutions, and other indications of inefficiency. Wasting time and effort may ultimately prove costly [S1] for ASD projects.

Lastly, *sprint cancellations* can imply wasted effort. Especially "unsystematic changes in requirements […] sometimes led to cancelled sprints" [S28, p. 10]. Moreover, dependencies with other projects and delayed customer requirements may cause sprint cancellations. In sum, "cancellation of sprints could easily result in degrading the performance and velocity of the project and delaying the product delivery" [S28, p. 16].

Regarding ***Planning and Estimation Problems***, we describe *Inadequate Effort / Time Estimation, Overcommitment, Inaccurate / Unrealistic / Limited Planning, Inadequate Long-Term Architecture-Planning* and *Difficult Progress Measuring.*

ASD can cause *inadequate effort and time estimates*. Most frequently, the team tended to overestimate its delivery capability [S24] and underestimated the required amount of work [S27,S59] required to deliver the features [S8], causing estimates to be unrealistic [S24] or inaccurate [S25]. Volatile requirements [S23,S28] and a lack of knowledge [S33,S51] were also reported as reasons for inadequate estimates.

As a related issue, *overcommitment* tends to occur in ASD teams [S41]. In this regard, both the team itself [S51] as well as "product management [that] pressurizes the development to create unrealistic plans" [S25, p. 32] can cause overcommitment. Hence, "sometimes […] too many stories are assigned to a […] sprint" [S52, p. 361], leading to unsuccessful sprint goal fulfillment.

Inadequate estimates also seem related to *inaccurate / unrealistic planning* as they cause unrealistic plans [S24] and hinder effective sprint planning [S52]. To add, changing requirements [S38], limited time, and large-scale projects [S49] also impede planning, which can "[affect] the team's chance of really committing to the sprint" [S51, p. 858].

Issues in *long-term architecture planning* suggest that planning beyond the next iterations may also be compromised, as architecture planning "has to take in consideration a lot of unknown upcoming variability" [S42, p. 8]. Inadequate architecture planning in the initial stages may lead to problems later on, i.e., when adding new clients to the system [S57], resulting in increased costs [S69] if rework is required.

Lastly, *difficulties in measuring progress* were identified. "Finding the right metrics" [S31, p. 9] appears challenging, as "these [metrics] are not always appropriate […] to agile projects" [S22, p. 6]. In addition, displaying progress in projects without a GUI is difficult [S33], however, this issue does not appear exclusively in ASD projects.

Table 7
Development process issues.

Issue Theme	Issue	Studies in our Sample	∑
Efficiency Problems	Reduced Productivity / Development Progress / Developer Performance	[8,12,13,15,16,18,23,27,28,33,34,37,46,49,50,53,59,66,67]	**19**
	Meeting Overhead	[1,13,33,43,46,48,50,51,53,60,64,65,68,70]	**15**
	Required Rework	[2,8,10,12,18,23,26,27,51,58,61,67]	**12**
	Distracting Workflow Interruptions	[8,46,53,64,65,66,70]	**7**
	Wasted Development Effort	[1,23,69]	**3**
	Context Switching / Task Switching	[23,34]	**2**
	Sprint Cancellation	[28]	**1**
Planning & Estimation Problems	Inadequate Effort / Time Estimation	[8,17,23,24,25,27,28,33,51,52,59,61]	**12**
	Inadequate / Inaccurate / Unrealistic / Limited Planning	[8,20,24,25,38,49,51,52]	**8**
	Inadequate Long-Term (Architecture) Planning	[7,24,42,50,58]	**5**
	Difficult Progress Measuring	[22,31,33,41,62]	**5**
	Overcommitment	[25,41,51,52]	**4**
Requirements Engineering Problems	Feature / Requirement / Task / Code Dependencies	[2,7,10,15,23,28,49,54,59,63,64]	**11**
	Inadequate Requirements Prioritization / Reprioritization	[7,15,24,27,38,50,51,58,59,61]	**10**
	Incomplete / Hidden / Late / Unclear / Conflicting / Vague / Overdefined / Ambiguous Requirements	[2,15,20,23,28,32,34,59,69]	**9**
	Negligence of Non-Functional Requirements / Quality Requirements	[2,8,10,12,15,18,23,26,61]	**9**
	Difficult Requirements Elicitation / Gathering	[8,27,34,51,57,60,63,67]	**8**
	Inadequate Requirement Specification / Separation	[2,8,15,46,51,57]	**6**
	Inadequate Requirements Assumptions	[1,2,18,27,67]	**5**

For ***Requirements Engineering Problems***, we categorize *Difficult Requirements Elicitation / Gathering, Inadequate Requirement Specification / Separation, Inadequate Requirements Assumptions, Various Requirements Issues, Feature / Requirement / Task / Code Dependencies, Inadequate Requirements Prioritization / Reprioritization,* and the *Negligence of Non-Functional Requirements / Quality Requirements.*

Several issues may occur during *requirements elicitation and gathering* in the RE process. Requirements gathering may prove difficult if the customer is not available [S27,S67], leading to "delays and loss of productivity" [S27, p. 527]. Additionally, working out "user requirements and quality of use in cooperation with [end users] of the product" [S63, p. 44] depicts a challenge.

Requirements specification and separation can also be flawed when product backlogs contain a "mixture of very detailed and very general items" [S51, p. 858]. Splitting "features into user stories small enough to be implemented in a two week sprint" [S57, p. 32] can thus be difficult. Also, ambiguous requirements can result when items are not adequately described [S15]. In addition, when "agile teams together with the customer fail to specify clearly the scope of [quality requirements], verifying the satisfaction of [quality requirements]" [S2, p. 279] becomes challenging.

Inaccurate requirements may result from *inadequate assumptions* made due to the unavailability of the customer and a lack of information [S1,S2,S18,S27,S67]. The impact of *inadequate assumptions* are mostly requirements that are misaligned with customer demands [S27,S67], requiring rework to meet the actual requirements [S18,S27].

Altogether, *various problems with requirements* can occur, such as incomplete [S32,S69], ambiguous [S2,S15,S20], vague [S34] and delayed requirements [S23,S28]. These issues cause delayed deliveries [S23,S59], sprint cancellations [S28], increased requirement volatility [S15], and unclear expectations for developers regarding required deliverables [S2,S34,S69].

The most frequent issue concerns *dependencies between features, requirements and code,* since when an "interdependency among user stories was found, it might have required significant refactoring with consequences for the whole structure of the software" [S7, p. 87]. Documenting dependencies can be challenging [S10], while they generally hinder the fulfillment of tasks [S28,S49,S63]. Most importantly, changes in dependent features, code, or requirements automatically lead to changes in other areas [S15,S17,S54].

Challenges may also arise during *requirements prioritization*, i.e., when requirements are not prioritized at all [S24] or not to a sufficient extent [S7,S59]. Non-functional requirements (NFR) are often not prioritized appropriately [S15], as in agile RE often "inadequate attention [is] given to non-functional requirements" [S61, p. 16].

Since the delivery of features has a higher priority, *NFRs* are frequently *neglected* [S2]. In this vein, "teams remove invisible [NFRs] from the PB to let [functional requirements] get higher priority" [S2, p. 278]. To add, in early project stages, "customers often focus on core functionality and ignore issues related to scalability, maintainability, portability, safety or performance" [S61, p. 464]. Neglecting NFRs can result in architecture technical debt [S15], rework [S26], and decreased system performance [S23].

4.6. Customer issues

In this section, we examine customer-related issues in ASD methodology use, including ***Customer Relationship Problems, Customer Behavior Problems,*** and ***Problematic Project Constraints*** defined by the customer.

In the theme on ***Customer Relationship Problems***, we classify *Lack of Trust between Team and Customer* and *Difficult Customer Communication*.

We found indications of a *lack of trust between the team and customers*, particularly when customers "come from the traditional development background and do not understand or trust the development" [S61, p. 466]. Additionally, it is "a challenge that stakeholders understand that the development team can make independent (detailed) decisions" [S63, p. 44]. Similarly, customers "harboured skepticism or misconceptions about Agile methods, leading to resistance towards collaboration" [S27, p. 525].

Moreover, *difficult customer communication*, i.e., when customers are unavailable, complicates the relationship [S26]. To add, communicating without direct engagement with the end-users may prove difficult. Hence, when "information originated from the customer has to flow through several different external and internal units before reaching to the development teams" [S15, p. 9], this information may be distorted.

The theme of ***Customer Behavior Problems*** includes issues such as *Inadequate Customer Access / Attendance / Availability / Responsiveness / Involvement, Lack of Customer Feedback, Demoralizing Customer Feedback, Volatile Requirements, Lack of Customer Knowledge,* and *Skepticism of Customers.*

The main concern here is *inadequate customer access and involvement,* since "onsite customer representation is difficult to attain" [S61, p. 45]. Generally, "customer [involvement] was seen as 'the most difficult part of Agile' and 'the biggest problem' because 'Agile [requires] fairly strong customer involvement'" [S27, p. 525]. Lacking involvement may result in a lack of critical information [S1], delayed decisions [S46], developer idle time [S59], and missing feedback leading to inadequate assumptions [S67].

The *lack of customer feedback* may prove especially problematic. As [S27] illustrates, "customer feedback is of vital importance in ensuring the desired product is being developed and delivered incrementally" (p. 527). Similarly, "when business feedback is insufficient, teams are not able to assess how the features meet the requirements" [S67, p. 2165], resulting in "delays and a loss of productivity" [S67, p. 2165]. In some cases, *demoralizing customer feedback* was perceived as discouraging, leading to a decline in performance by developers who took the feedback personally, [S59]. Nevertheless, such negative feedback was also important for making progress.

While feedback is essential, "feedback provided by the customers on latest features produced in a sprint [introduces] new requirements or changes in existing requirements" [S28, p. 10], as clients' demands frequently change [S15,S28,S51,S69]. While some *requirement volatility* is expected since "not all of them are fixed at the beginning" [S63, p. 44], "unsystematic changes in requirements […] sometimes led to cancelled sprints" [S28, p. 10]. Further consequences may arise, including planning and estimation difficulties [S8,S61], difficult NFR management [S15], communication lapses [S26], delivery delays [S23], and wasted effort in requirements gathering [S69].

Lastly, with a *lack of customer knowledge*, "unrealistic expectations become more likely […] as the customer does not know what is possible and what is not" [S9, p. 79]. Moreover, when customers are not familiar with the system and business processes [S27], unrealistically high demands for expedited delivery can emerge [S9].

Finally, we examine ***Problematic Project Constraints*** defined by the customer, which include *Tight Deadlines / Time Pressure, Tight Budget Constraints / Fixed Price Contracts,* and *Unrealistic Customer Expectations.*

The iterative delivery of software implies *tight deadlines and time pressure*, causing a decline in documentation [S10,S52] and learning opportunities [S4,S34]. To add, technical debt begins to accumulate [S12,S18,S42] and developers may feel exhausted due to the constant pressure [S46,S68]. In extremely short sprints, the completion of tasks may sometime not be feasible [S52], particularly due to the high amount of time spent in meetings [S53].

Also, *budgetary constraints,* including fixed-price contracts pose challenges for ASD projects. Generally, "with Agile it's difficult to do fixed price projects. Agile talks about embracing change, [you] can't do fixed price projects with changes coming in" [S27, p. 526]. Due to limited budgets, also some practices may be omitted, i.e., refactoring and pair programming [S40], which can lead to technical debt [S42].

Customers may also have *unrealistic expectations* regarding the project scope and delivery time [S9]. Clients with technical expertise appear

Table 8
Customer issues.

Issue Theme	Issue	Studies in our Sample	∑
Customer Relationship Problems	Difficult Customer Communication	[14,15,26,59,60]	**5**
	Lack of Trust between Team & Customer	[9,13,61,63]	**4**
Customer Behavior	Inadequate Customer Access / Attendance / Availability / Responsiveness / Involvement	[1,13,17,18,20,22,24,26,27,46, 54,59,61,63,66,67,68]	**17**
	Volatile Requirements	[8,15,17,23,26,27,28,48,51,61,63,67,69]	**13**
	Lack of Customer Feedback	[2,27,28,50,59,61,67]	**7**
	Lack of Customer Knowledge	[9,21,27,69]	**4**
	Skepticism of Customers	[21,27]	**2**
	Demoralizing Customer Feedback	[59]	**1**
Problematic Project Constraints	Tight Deadlines / Time Pressure	[4,7,10,12,18,20,34,40,42,46,52,53,57,58,59,68]	**16**
	Tight Budget Constraints / Fixed Price Contracts	[27,40,68,69]	**4**
	Unrealistic Customer Expectations	[9,20]	**2**

less problematic, while with less experienced clients, "their expectations are mostly too high and need to be brought down to earth first" [S9, p. 79]. Additionally, [S20] reports "inappropriate assumptions about [the] project scope made by [the] client, due to the development team's flexible agile-related approach" (p. 14), which underlines the need for sufficient customer knowledge.

4.7. Technical issues

This chapter discusses the ***Incorrect Application of ASD practices***. We define *Insufficient / Inadequate / Limited Documentation, Negligence of Refactoring / Improvement, Lack of Discipline in Agile Environment, Lack of Formal Guidelines, Injection of new Requirements during Sprints,* an *Ambiguous / Missing Definition of Done, Low Test Coverage / Inefficient Testing* as well as *Ineffective Retrospective Activities* to be part of this theme.

Insufficient / inadequate / limited documentation is a common issue in ASD. It appears that the "time constraint has always been a factor in having less documentation because it is the first thing you cut if you do not have time" [S10, p. 7]. The consequences can include loss of knowledge in case of staff fluctuations [S6,S54], accumulation of technical debt [S10], progress tracking [S64], and difficult onboarding of new members [S18]. Contrastingly, [S19,S21,S28] describe exorbitant documentation, especially when demanded from upper management, which indicates a lack of trust [S21,S22]. Requiring the "teams to submit relatively heavy documentation such as frequent status reports" [S28, p. 11] contrasts "a light-weight process involving minimum documentation" [S28, p. 11], which may affect performance. Hence, "wrongly [understanding ASD] as 'no documentation at all'" [S54, p. 72] and overdocumentation may imply negative consequences.

Due to time pressure, a *negligence of refactoring / improvement* was observed. Teams often neglect refactoring due to the need to deliver new features [S51,S53], which accumulates technical debt [S42] that requires rework or inevitable rewriting of the software [S12,S61]. Besides tight deadlines, budget constraints [S40] and penalties for missed deadlines [S42] may affect the amount of refactoring conducted by the team. Neglecting refactoring may also increase the development costs due to rework [S10,S61].

Several studies report that "the time pressure inherent to higher-workload periods *reduces discipline*" [S53, p. 1437]. To add, as the ASD "environment does not have a lot of rules" [S12, p. 12], developer behavior during coding work [S12] or the application of practices [S60] may deteriorate. Still, only a few ASD practices provide guidelines that developers can adhere to [S52].

The *lack of formal guidelines* was frequently deemed problematic [S5, S52,S62], as ASD may be perceived as "too vague to provide guidance" [S5, p. 66], without directives "that can guide you to make the best use out of agile" [S52, p. 363]. Therefore, "the result of applying Scrum will always be a unique version" [S52, p. 363].

One existing guideline pertains to the stability of requirements during a sprint [S60], which is frequently ignored due to the *injection of new requirements*. As such, "requirement changes […] could occur at any time in the project as customers frequently changed their minds" [S28, p. 10]. The inclusion of new requirements can result in rising workloads and decreases in quality [S55]. An iteration that was cancelled due to new requirements also appears to have "impacted the overall delivery plan" [S28, p. 10].

Generally, Scrum projects have a *'definition of done'* that indicates the completion of a task. Nevertheless, certain projects either *lack* this *definition* or *misinterpret* it [S36], causing misunderstandings [S8,S36]. Similarly, lacking this definition can make it difficult to "complete what was planned for the current iteration" [S51, p. 862].

In analogy to refactoring, *low test coverage / inefficient testing* can occur due to tight schedules. According to [S52], regression testing, "test-driven development or unit testing" (p. 3612) appeared not feasible due to tight deadlines. [S8,S58] describe low test coverage and lengthy test times, while continuous testing also depicts a challenge.

Lastly, *retrospective meetings* may be conducted *ineffectively*. [S43, S44] describe that these activities can be unprepared, too repetitive, cause a blame / complain game or are only focused on negative aspects

Table 9
Technical issues.

Issue Theme	Issue	Studies in our Sample	∑
Incorrect Application of ASD Practices	Insufficient / Inadequate / Limited Documentation	[6,8,10,15,18,19,20,21,22,26,28,52,54,55,61,64]	**16**
	Negligence of Refactoring / Improvement	[3,12,40,42,51,53,61]	**7**
	Injection of new Requirements during Sprints	[14,28,55,60,69]	**5**
	Lack of Discipline in Agile Environment	[12,41,53,55,60]	**5**
	Lack of Formal Guidelines	[5,52,62]	**3**
	Ambiguous / Missing Definition of Done	[8,36,51]	**3**
	Low Test Coverage / Inefficient Testing	[8,52,58]	**3**
	Ineffective Retrospective Activities	[43,44]	**2**

[S44]. As a consequence, these issues resulted in "meetings running much longer than planned, finishing without clear action items to improve the next sprint or not discussing problems that existed during the sprint" [S43, p. 6232].

4.8. Organizational context issues

In this section, we address issues in the organizational context. We divided these issues into ***Inadequate Support for ASD*** and ***Inadequate Environments for ASD***, as reported in Table 10.

Inadequate Support for ASD includes *Unsuitable Corporate Culture for Agile Methods, Resistance to Change, Difficult Mindset Change, Inadequate Introduction of Agile Methods, Lack of Agile Training, Lack of Management Support for Agile Methods, Lack of Experienced Agile Developers* and *Staff Fluctuation.*

The *inadequacy* of the *corporate culture* represents a significant issue, as changing the culture "from a command and control/mechanistic worldview to a future of autonomous, self-managed agents in a systemic organisation" [S21, p. 10] is a challenging task [S51,S54,S67]. Since "a highly-agile team will not fit in well with a heavyweight process-oriented organization that prefers planning and formal communication" [S69, p. 352], this cultural change is important for ASD projects.

Resistance to change and *difficult mindset change* are closely related issues. Akin to the cultural transformation, the mindset of the developers needs to change, which is considered difficult [S28]. As [S22] stated, "agile is more than a set of practices used by IT requiring wide-ranging change to work patterns" (p. 5). With the presence of *resistance to change*, it may thus be "impossible to drive the change from bottom up" [S57, p. 23]. Particularly "traditional practitioners may fear and resist the change given control moves to the team [in the sense of self-organising teams]" [S22, p. 5]. Consequently, ASD may be abandoned when change resistance is too strong [S60].

A proper introduction of ASD is necessary, however, as several studies reported that "agile methods were introduced quickly and thus the education and training was inadequate in the beginning" [S54, p. 73], implying an *inadequate introduction.* Specifically, "[communicating] the method clearly to the team members" [S6, p. 204] appears important, since "not having a clear understanding of what they are doing will attract more resistance from the team members" [S6, p. 204].

Additionally, a *lack of agile training and coaching* was observed. Individuals may not feel prepared to manage the methodology [S54] when a "lack of training in specific practices or techniques" [S22, p. 6] exists, despite people "[needing] to learn new skills in addition to their earlier area of responsibilities" [S54, p. 73].

A further key issue is *lacking management support for ASD methodologies*, which can impede adoption [S6,S22,S28,S55,S62]. The consequences include difficulties in practicing agile [S6], failed implementation of the methods [S55] or micromanagement when trust is lacking [S21].

Lastly, we discuss issues that concern human resources. Generally, ASD "requires skilled, self-directed and motivated team players" [S22, p. 6]. Contrastingly, a *lack of experienced agile developers* was reported [S13, S21], which may decrease the quality of the delivered software product [S18] and cause a decline in customer trust [S9]. As reasons for the lack of skilled agile developers, [S13] cites the shortage of university programs that teach ASD and missing "agile-specific recruitment policies" [S13, p. 55].

Finally, *staff fluctuation* can aggravate the situation in organizations that employ ASD. While staff fluctuation is not an agile-specific problem, its effects seem particularly noticeable in ASD. As such, when "people leave, without documentation the knowledge is lost" [S54, p. 72]. This loss of knowledge can have negative effects on a project [S15], especially when "the knowledge resides in only one person" [S6, p. 202]. Additionally, decreases in productivity [S16] and delays can occur [S8]. Lastly, productivity losses can also result from new members joining who need time to get into a rhythm [S16,S19].

For ***Inadequate Environments for ASD,*** we address *Agile in a Distributed Setting,* the *Use of Agile in a Scaled Fashion,* and *Agile in Non-Agile Environments*.

The *distribution of ASD teams* can imply communication difficulties [S15,S54], collaboration issues [S19,S54], difficulties in establishing a team spirit [S57], coordination problems [S64] and the degradation of practices [S60]. Geo-temporal differences also make scheduling challenging [S60,S64]. Technical issues with tools to overcome the distance also represent a source for meeting overhead [S64,S65]. Finally, distribution can affect development performance, as developers find it difficult to establish trust [S15].

Several issues also arose when *ASD* was applied in a *scaled fashion*. Coordination is particularly complicated by a large amount of personnel with dependencies between projects [S5,S49,S58], causing coordination costs [S5]. Meetings are also negatively affected by larger teams [S33], while productivity decreases were observable [S23]. Additionally, requirements handling can be hampered [S10,S58] when many people are involved. Lastly, difficulties in planning and estimation can occur due to the complexity of large-scale scenarios [S24,S58]. As such, due to "the size and complexity of the large-scale project, it seems impossible to plan everything that needs to be done" [S49, p. 7002].

The last issue concerns the use of *agile in a non-agile environment* [S1, S21,S22]. Working with a non-agile customer was reported as difficult in regard to communication [S21,S59,S69], since "communication requirements for agile [teams are] not met by management/non-agile teams" [S21, p. 20]. Moreover, an unexperienced customer that "does not buy into the agile mind set […] greatly reduces the team's ability to be agile" [S69, p. 352]. In addition, the organizational environment can be problematic in case it "is not homogeneous in terms of way-of-working, or when some parts of the organization/teams follow agile methods while others do not" [S1, p. 11].

Table 10
Organizational context issues.

Issue Theme	Issue	Studies in our Sample	∑
Inadequate Support for ASD Methodologies	Unsuitable Corporate Culture for Agile Methods	[6,19,21,22,48,51,54,62,66,67,69]	**11**
	Staff Fluctuation	[6,8,15,16,18,19,40,51,54,61]	**10**
	Resistance to Change	[6,21,22,31,40,57,60,62]	**8**
	Lack of Management Support for Agile Methods	[6,21,22,28,51,55,62]	**7**
	Inadequate Introduction of Agile Methods	[6,19,21,38,54]	**5**
	Lack of Experienced Agile Developers	[9,13,18,21,22]	**5**
	Lack of Agile Training	[22,31,54,62]	**4**
	Difficult Mindset Change	[21,22,28]	**3**
Inadequate Environments for ASD Methodologies	Use of Agile in a Distributed Setting	[10,13,15,19,21,22,24,28,31,54,57,60,62,64,65]	**15**
	Use of Agile in Scaled Fashion	[5,10,21,22,23,24,33,49,58,62,64]	**11**
	Agile in Non-Agile Environments	[1,21,22,23,31,59,69]	**7**

4.9. Product and project outcome issues

Lastly, we outline issues pertaining to the product and project outcomes, including ***Quality Problems, Architecture Problems*** and ***Project Outcome Problems***. Table 11 contains the issues associated with this dimension.

For ***Quality Problems***, we discuss *Reduced Product Quality / Performance, Technical Debt, Poor Code Structure, Difficult Requirements Meeting / Validation, and Difficult Introduction of New Features*.

Several issues concern the *product quality*, i.e., lower internal software and systems quality [S25], higher data retrieval time [S2], affected security [S61], while also "software performance, stability, testability, and maintainability" [S23, p. 5] declined. In addition, more software bugs were detected [S55,S56]. As key factors, studies illustrated inadequate documentation [S10], negligence of NFRs [S23], pressure on developers to deliver [S52], and technical debt [S10,S18,S25] due to insufficient refactoring [S53].

Technical debt is another frequently described issue. Generally, "incurring technical debt brings short-term benefits such as reduced development time or effort" [S18, p. 10]. However, the long-term effects include reduced quality and performance, increased development duration [S10], or rework and rewriting of the software [S61]. As potential reasons for the occurrence of technical debt, the studies report "missing and outdated […] documentation in ASD" [S10, p. 9], lack of discipline [S12], continuous development [S21], and, most frequently, the pressure to deliver features quickly [S12,S18,S25,S53].

Similarly, several studies reported a *poor code structure* of the software [S2,S61], requiring rework to be fixed [S61]. *Poor code structure* was also found to hinder the introduction of new features, demanding amendments to enable their implementation [S23,S42].

The literature also reports instances where ASD *fails to meet customer requirements*. The lack of customer feedback on requirements may cause inadequate assumptions that lead to faulty features [S18,S27,S67], as "teams were unable to assess how well the features met the requirements" [S27, p. 527]. Also, inadequate documentation may result in basic features not being implemented in the software [S10].

For the related ***Architecture Problems***, we describe *Inadequate Architecture* and *Architecture Technical Debt*.

Several studies reported that the *architecture* became *inadequate* as the project progressed [S2,S61]. Since not all requirements are known in the beginning [S2], "challenging requirements lead to a complex architecture" [S69, p. 350]. As [S69] points out, "with each iteration, the team has to consider if the existing architecture is suitable for the current set of requirements being implemented (p. 351). Having to rework the architecture may incur additional costs, while it often does "not completely address the problem of inadequate or inappropriate architecture" [S61, p. 464].

Furthermore, *architectural technical debt* was frequently reported. Similar to technical debt, "inconsistencies due to shortcuts and the low-prioritization of the necessary refactoring" [S42, p. 9] are antecedents of architectural technical debt. Architecture debt is considered the most difficult deficit to fix [S12], "since changing system architecture requires lots of cooperation" (p. 13). Overall, "due to the continuous and inevitable accumulation of [architecture technical debt] and the impossibility of removing it all", [S42, p. 17], this issue illustrates a very specific challenge of ASD.

Lastly, we address ***Project Outcome Problems*** including *Delivery Delays, Increased Cost,* and *Dissatisfied Customers*.

Several studies described *delivery delays*, caused by technical debt that requires fixing [S18], inadequate effort estimates [S8], unresponsive customers [S1,S67], volatile requirements [S8,S23,S28] and rework required to add new features [S51].

We also identified instances where the budget was exceeded and *increased cost* resulted. The most notable cause was required rework [S61,S67] to meet customer requirements [S27]. Readjusting the architecture analogously contributed to rising costs [S69]. In addition, wasting effort on the wrong products also implies wasted costs [S1].

Lastly, we found two instances of *dissatisfied customers* [S9,S10]. The reasons were high customer expectations that could not be met [S9] and NFRs that were not implemented in the final product [S10]. While we only identified two cases, we may assume that dissatisfaction may also have occurred in other instances, considering the large variety of outcome-related issues identified in our analysis.

4.10. Relationships between identified issues in ASD methodology use

Several relationships between the identified issues can be found in the literature sample. To illustrate the complexity of the dark side of ASD, we created a relationship diagram in Fig. 12. The diagram includes 65 relationships among issues that were described in at least two separate publications. We also indicate issues that were most commonly impacted by other antecedent problems (dotted red line). Analogously, we identify issues that may especially lead to further negative consequences by preceding other problems (solid red line). Issues fulfilling the criteria for highlighting either influenced at least three other issues or were at least influenced by three other problems. Additionally, the diagram highlights the frequency of observed relationships, as illustrated by the varying **strength** of the arrows between issues. In Tables 12 and 13, we detail 65 relationships from the literature and list corresponding contributions, while the online appendix features further illustrative quotes.

Referring to Tables 12 and 13, the ***most influential issues*** are *inadequate customer involvement, tight deadlines and time pressure, iterative delivery pressure for developers, reporting pressure, meeting overhead, various RE-related issues, inadequate architecture, technical debt, required rework,* and *staff fluctuation*. The issues caused by the ***customer*** seem particularly adverse, as they precede numerous other issues. *Inadequate customer involvement* can be especially harmful in ASD, as difficulties in "getting requirements from the business stakeholders [result] in unnecessary delays and loss of productivity" [S67, p. 2164]. Moreover, fluctuations in business involvement cause significant changes in team productivity [S18,S59,S66,S67]. Most importantly, if the client is not

Table 11
Product and project outcome issues.

Issue Theme	Issue	Studies in our Sample	∑
Quality Problems	Reduced Product Quality / Performance	[2,10,18,23,25,34,52,55,56,61,70]	**11**
	Technical Debt	[10,12,18,21,25,51,53,57,61]	**9**
	Difficult Requirements Meeting / Validation	[5,10,18,23,27,67,68]	**7**
	Difficult Introduction of New Features	[5,23,42]	**3**
	Poor Code Structure	[2,25,61]	**3**
Architecture Problems	Inadequate Architecture	[2,15,57,61,69]	**5**
	Architecture Technical Debt	[12,25,42]	**3**
Outcome Problems	Delivery Delays	[1,8,10,18,23,27,28,49,51,59,60,67,68,70]	**14**
	Increased Cost	[1,8,10,27,61,67,68,69]	**8**
	Dissatisfied Customer	[9,10]	**2**

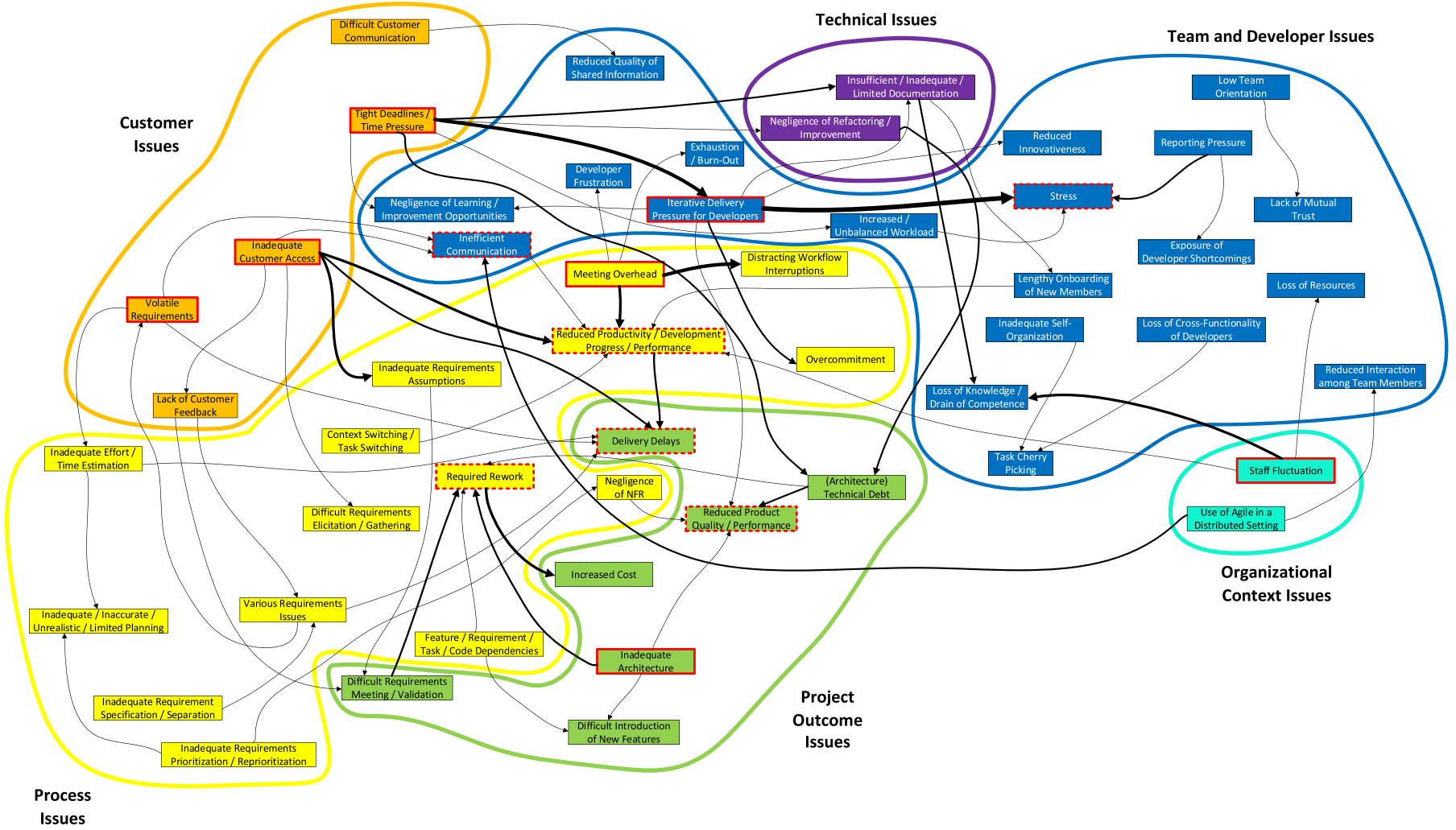


Fig. 12. Relationship Diagram on Related Issues in ASD Methodologies.

Table 12
Relationships (customer, developer & team and technical issues).

Driving Issue	Issue Theme	Affected Issue	Issue Theme	Authors	∑
Difficult Customer Communication	**Customer-related Issues**	Reduced Quality of Shared Information	Developer & Team Issues	[15,59]	**2**
Inadequate Customer Access / Attendance / Availability / Responsiveness / Involvement		Lack of Customer Feedback	Customer-related Issues	[27,67]	**2**
		Difficult Requirements Elicitation / Gathering	Process Issues	[27,67]	**2**
		Inadequate Requirements Assumptions	Process Issues	[1,18,27,67]	**4**
		Reduced Productivity / Development Progress / Developer Performance	Process Issues	[18,59,66,67]	**4**
		Inefficient Communication	Developer & Team Issues	[61,68]	**2**
		Delivery Delays	Product & Outcome Issues	[1,67,68]	**3**
Lack of Customer Feedback		Incomplete / Hidden / Late / Unclear / Conflicting / Vague / Overdefined / Ambiguous Requirements	Process Issues	[2,28]	**2**
		Difficult Requirements Meeting / Validation	Product & Outcome Issues	[27,67]	**2**
Tight Deadlines Time Pressure		Insufficient / Inadequate / Limited Documentation	Technical Issues	[10,52,61]	**3**
		Negligence of Refactoring / Improvement	Technical Issues	[40,42]	**2**
		Increased / Unbalanced Workload	Developer & Team Issues	[53,68]	**2**
		Iterative Delivery Pressure for Developers	Developer & Team Issues	[4,29,46,52,59]	**5**
		Negligence of Learning / Improvement Opportunities	Developer & Team Issues	[4,34]	**2**
		Technical Debt	Product & Outcome Issues	[12,18,57]	**3**
Volatile Requirements		Inadequate Effort / Time Estimation	Process Issues	[28,61]	**2**
		Inefficient Communication	Developer & Team Issues	[26,61]	**2**
		Delivery Delays	Product & Outcome Issues	[8,23]	**2**
Inadequate Self-Organization	**Developer & Team Issues**	Task-Cherry-Picking	Developer & Team Issues	[13,66]	**2**
Increased / Unbalanced Workload		Stress	Developer & Team Issues	[37,38]	**2**
Inefficient Communication		Reduced Productivity / Development Progress / Developer Performance	Process Issues	[13,49]	**2**
Iterative Delivery Pressure for Developers		Insufficient / Inadequate / Limited Documentation	Technical Issues	[52,64]	**2**
		Overcommitment	Process Issues	[25,27,51]	**3**
		Negligence of Learning / Improvement Opportunities	Developer & Team Issues	[4,29]	**2**
		Reduced Innovativeness & Creativity	Developer & Team Issues	[3,4]	**2**
		Stress	Developer & Team Issues	[4,5,11,29,46,52]	**6**
		Reduced Product Quality / Performance	Product & Outcome Issues	[52,55]	**2**
Lengthy Onboarding of New Members		Reduced Productivity / Development Progress / Developer Performance	Process Issues	[16,18]	**2**
Loss of Cross-Functionality of Developers		Task-Cherry-Picking	Developer & Team Issues	[28,66]	**2**
Low Team Orientation		Lack of Mutual Trust	Developer & Team Issues	[48,50]	**2**
Reporting Pressure		Exposure of Developer Shortcomings	Developer & Team Issues	[41,65]	**2**
		Stress	Developer & Team Issues	[46,52,64]	**3**
Insufficient / Inadequate / Limited Documentation	**Technical Issues**	Lengthy Onboarding of New Members	Developer & Team Issues	[18,61]	**2**
		Loss of Knowledge / Drain of Competence	Developer & Team Issues	[6,8,54]	**3**
Negligence of Refactoring / Improvement		Architecture Technical Debt	Product & Outcome Issues	[12,42]	**2**
		Technical Debt	Product & Outcome Issues	[12,53,61]	**3**

responsive, the team may make assumptions in the RE process due to their absence [S27,S67], which may cause deviations from the actual requirements [S27]. Also, when customers do not dedicate enough time to feedback and involvement, delivery delays may result [S1,S67,S68], as *volatile requirements* need to be clarified first [S8,S23]. Similarly, *reduced productivity* due to the *unavailability of the customer* contributes to *delivery delays* [S28,S46,S67], as featured in Fig. 12.

Furthermore, developers are affected by *tight deadlines* defined by the customer and the accompanying *time pressure*, thus impeding the correct application of ASD practices. Due to the *iterative delivery pressure for developers* that especially results from short timeframes between iterations [S4,S29,S46,S52,S59], developers often lack time to properly

Table 13
Relationships (process, product & outcome, organizational context issues).

Driving Issue	Issue Theme	Affected Issue	Theme of Issue	Authors	∑
Context Switching / Task Switching	**Process Issues**	Reduced Productivity / Development Progress / Developer Performance	Process Issues	[23,34]	**2**
Feature / Requirement / Task / Code Dependencies		Required Rework	Process Issues	[7,23]	**2**
		Difficult Introduction of New Features	Product & Outcome Issues	[23,54]	**2**
Inadequate Effort / Time Estimation		Inadequate / Inaccurate / Unrealistic / Limited Planning	Process Issues	[24,52]	**2**
		Delivery Delays	Product & Outcome Issues	[8,59]	**2**
Inadequate Requirement Specification / Separation		Incomplete / Hidden / Late / Unclear / Conflicting / Vague / Overdefined / Ambiguous Requirements	Process Issues	[2,15]	**2**
Inadequate Requirements Assumptions		Difficult Requirements Meeting / Validation	Product & Outcome Issues	[27,67]	**2**
Inadequate Requirements Prioritization / Reprioritization		Inadequate / Inaccurate / Unrealistic / Limited Planning	Process Issues	[24,38]	**2**
		Negligence of Non-Functional Requirements / Quality Requirements	Process Issues	[15,61]	**2**
Incomplete / Hidden / Late / Unclear / Conflicting / Vague / Overdefined / Ambiguous Requirements		Volatile Requirements	Customer-related Issues	[15,69]	**2**
		Delivery Delays	Product & Outcome Issues	[23,59]	**2**
Meeting Overhead		Distracting Workflow Interruptions	Process Issues	[46,53,64,65,70]	**5**
		Reduced Productivity / Development Progress / Developer Performance	Process Issues	[46,50,53,64]	**4**
		Developer Frustration	Developer & Team Issues	[46,64]	**2**
		Exhaustion / Resource Depletion / Burnout	Developer & Team Issues	[46,65]	**2**
Negligence of Non-Functional Requirements / Quality Requirements		Reduced Product Quality / Performance	Product & Outcome Issues	[23,61]	**2**
Required Rework		Increased Cost	Product & Outcome Issues	[23,27,61,67]	**4**
Reduced Productivity / Development Progress / Developer Performance		Delivery Delays	Product & Outcome Issues	[28,46,67]	**3**
Difficult Requirements Meeting / Validation	**Product & Outcome Issues**	Required Rework	Process Issues	[18,27,67]	**3**
Inadequate Architecture		Required Rework	Process Issues	[2,57,61]	**3**
		Difficult Introduction of New Features	Product & Outcome Issues	[25,57]	**2**
		Reduced Product Quality / Performance	Product & Outcome Issues	[2,61]	**2**
Technical Debt		Required Rework	Process Issues	[10,12]	**2**
		Reduced Product Quality / Performance	Product & Outcome Issues	[10,18,25]	**3**
Staff Fluctuation	**Organizat-ional Context Issues**	Loss of Knowledge / Drain of Competence	Developer & Team Issues	[6,15,19,54]	**4**
		Loss of Resources	Developer & Team Issues	[8, 51]	**2**
		Reduced Productivity / Development Progress / Developer Performance	Process Issues	[16,19]	**2**
Use of Agile in a Distributed Setting		Inefficient Communication	Developer & Team Issues	[15,54,60]	**3**
		Reduced Interaction among Team Members	Developer & Team Issues	[15,54]	**2**

execute ASD practices, such as *refactoring* [S40,S42] and *documentation of development work* [S10,S52,S61,S64]. These crucial tasks are "the first thing you cut if you do not have time" [S10, p. 7] and may hamper *product quality* [S52,S55]. Overall, ***technical issues*** may especially prove costly in the long-term with the accumulation of *technical debt* reinforced by the *negligence of refactoring* [S12,S18,S42,S53,S57,S67] and the *loss of knowledge* due to *insufficient documentation* [S6,S8,S54]. To add, the latter also makes *onboarding new members* challenging [S18,S61], as the "velocity and quality tend to drop with the joining of a new team member" [S18, p. 11].

Under *tight deadlines* and *delivery pressure,* the *negligence of learning and improvement opportunities* is also more likely [S4,S29,S34]. In analogy to neglecting key practices stated before, it appears "difficult to get time for competence development [due to the] very tight time schedule" [S4, p. 527]. Being under time pressure to deliver software also leads to *increased workloads* [S53,S68] and consequently *stress* for the team [S4, S5,S11,S29,S46,S52]. As pointed out in [S5], teams are "under constant pressure to deliver and the presence of short feedback loops within each work iteration increased the time pressure and the debilitating effects of *stress* that accompanied it" (p. 68). Consequently, "failure to account for [stress] may have long-term detrimental effects on developer health" [S11, p. 574].

Various issues besides *iterative delivery pressure* appear to aggravate the situation for the ***development team***. Akin to pressure stated before, *reporting* represents a similar source of *stress* [S46,S52,S64]. This seems further corroborated by the *exposure of developer shortcomings* that become especially visible in daily meetings [S41,S65], with developers feeling "uncomfortable about demonstrating [their] work, because not many people could recognize any improvements" [S41, p. 22]. Moreover, *inadequate self-organization* [S13,S66] and lacking *cross-functionality of developers* [S28,S66] provoke *task-cherry-picking*. In this vein, "some tasks did not get picked, even when it had the highest priority" [S66, p. 154]. This situation could imply delivery delays if important tasks are not fulfilled at the end of an iteration. Overall, *inadequate customer behavior, pressure to deliver and report,* and *inadequate task allocation* can have consequences for the SD team and its performance, while also affecting the long-term health of projects.

As outlined in Table 13, several issues may occur in the ***process*** that have negative implications for SD projects. Most notably, *meeting overhead* associated with frequent ceremonies in ASD implies *distracting workflow interruptions* [S46,S53,S64,S65,S70], leading to *reductions in productivity and development performance* [S46,S50,S53,S64]. [S46] illustrates this relationship, since meetings "reduce the amount of time available to work on the deliverables" (p. 99). Attending too many meetings can also result in *frustration* [S46,S64], since "lengthy meetings impacted their ability to deliver as agreed" [S46, p. 102]. A high number of meetings also seems related to "feelings of fatigue and subjective workload" [S65, p. 29], thus implying *exhaustion* for developers [S46, S65].

By comparing Tables 12 and 13 and examining Fig. 12, it seems evident that the issues described before primarily aggravated the situation of the ***development team***. Simultaneously, a large proportion of issues in the ***SD process*** can particularly impair the *process performance* and affect the ***outcomes***. Especially issues related to RE and requirements management seem to give rise to further problems or could worsen present problems. Particularly *inadequate requirements specification* and *requirement prioritization* can prove problematic. *Inadequate specification* causes *incomplete, unclear* or *vague requirements* [S2,S15], which introduce *requirement volatility* [S13,S59] and *delivery delays* [S23, S59]. To add, when *requirements* are *prioritized inadequately*, planning becomes difficult "because of surprises" [S38, p. 284]. More critically, inadequate prioritization often entails the *negligence of NFRs*, ultimately affecting *product quality* [S23,S61]. Neglecting NFRs can also have long-term consequences, i.e., deficiencies in "software performance, stability, testability and maintainability" [S23, p. 5]. Lastly, "overlooking architectural aspects when prioritizing requirements accumulates architectural technical debt" [S15, p. 10], which requires *rework* to be fixed [S10,S12].

Similar ***shortcomings*** in the ***product*** besides *technical debt*, such as *inadequate architectures* or *failing to meet customer requirements,* generally contribute to *reduced product quality* [S2,S10,S18,S25,S61]. All these issues also demand *rework* to be fixed [S2,S10,S12,S18,S27,S28,S61, S67], which substantially adds to *increased development cost* [S23,S27, S61,S67], see Fig. 12. In particular, "rework of the architecture may add significantly to project cost" [S61, p. 463], with *technical debt* similarly causing "system quality and performance degradation […], increased development time […], increased maintenance costs […] and rework" [S10, p. 9]. To add, deviations from the actual requirements caused by assumptions made due to the absence of the customers ultimately require rework, "which [incurs] additional costs for the customers" [S27, p. 528]. In sum, deficiencies in the ***SD process***, especially in RE, and the ***project outcomes*** likely give rise to *rework* and *delivery delays*, ultimately also causing significant *cost increases.*

Lastly, ***organizational context issues*** such as *using agile in distributed settings* and *staff fluctuations* predominantly harm SD teams and their performance. Although *staff fluctuation* is not an ASD-exclusive issue, the *inadequate documentation* in ASD [S6,S8,S54] aggravates the *loss of knowledge* when team members leave [S6,S15,S19,S54], which also causes *productivity losses* [S16,S19]. *Distributed agile* as a rather inadequate environment for ASD can meanwhile cause *reduced interaction among team members* [S15,S54], while "language barriers and cultural differences" [S15, p. 9] may further result in *inefficient communication* [S15,S54,S60]. In sum, problems caused by the customer and problems occurring in the SD process often imply significant further issues for the SD team and the long-term performance of the projects.

The ***most frequently affected issues*** include *inefficient communication, stress, reduced product quality and performance, required rework, delivery delays,* and *reduced productivity and developer performance*. The issues that arise as a result of the preceding problems are primarily related to the ***SD process*** and the ***project outcomes***. While some of the above-mentioned issues were briefly discussed in the description of the most influential issues, especially *productivity and developer performance* can be aggravated by several preceding issues. Losses in productivity are particularly caused by *inadequate customer involvement* [S18,S59,S66, S67], which can "paralyze the work of agile teams" [S18, p. 12], as they experience idle time due to absent customers [S59]. Excessive time spent in meetings also impedes progress-making for ASD teams, since these *meetings* often imply significant *overhead* [S46,S50,S53,S64] and *distract* developers in their *workflow* [S46,S53,S64,S65,S70]. Other factors affecting *productivity* include *staff fluctuation* [S16,S19] and *onboarding of new members* [S16,S18], with teams experiencing "reduced productivity due to staff turnover" [S16, p. 418]. Furthermore, *delivery delays* are often a consequence of *reduced team productivity* [S28,S46,S67] and become more likely under *lacking customer involvement* [S1,S67,S68]. As shown in Fig. 12, *issues* with *requirements* pose further risks for *delivery delays. Volatile requirements* [S8,S23] or *unclear and incomplete requirements* jointly complicate *time and effort estimation* [S28,S61], which introduce "uncertainty in effort estimates and are perceived to be a direct cause of delays" [S23, p. 6].

Considering the *deterioration of product quality* and the *need for rework*, several product-related issues need to be considered. Most notably, *inadequate product architectures* [S2,S61] and *technical debt* [S10,S18,S25] cause a *decline in product quality and performance*. Especially technical debt can "result in unexpected project delays and lower software quality in the long run" [S18, p. 10]. To add, *neglecting NFRs* and developing *inadequate architectures* result in declines in "software performance, stability, testability, and maintainability" [S23, p. 5]. Both *technical debt* [S10,S12] and *inadequate architectures* [S2,S57,S61], together with other issues also increase the *need for rework*. Similarly, *failing to meet customer requirements* also *requires rework* [S2,S57,S61], especially under "total disregard of involvement" [S67, p. 2165] by the customer. Finally, *rework required* due to the reasons stated before and featured in Fig. 12 ultimately leads to *increased costs* in SD projects [S23, S27,S61,S67].

While the aforementioned issues mainly pertain to the ***SD process*** and ***the product***, also several concerns related to the ***team and developers*** appear preceded by other problems. *Stress* needs to be emphasized in this regard, as *stress* is primarily induced by the *iterative delivery pressure* [S4,S5,S11,S29,S46,S52], and *increased workloads* [S37, S38]. This relationship represents the most frequently identified cause-effect relation in our study, thereby emphasizing the "generally more stressful" [S52, p. 361] nature of ASD methodologies. This appears

further supported by *stress* experienced by developers "having to attend a meeting every day to report […] progress" [S52, p. 361]. Lastly, *team communication* can be *inefficient* in *distributed settings* [S15,S54,S60], as it appears "difficult to organize the agile development remotely" [S54, p. 77]. Also, the "non-availability of appropriate customer representatives" [S61, p. 467] was reported to cause *communication inefficiencies* [S61, S68].

Taken together, various issues occurring in ASD methodologies may give rise to further potential consequences or aggravate existing issues in a reciprocal interplay, as can be inferred from Fig. 12. This interplay will be further addressed in the subsequent sections, as it provides significant insights into *how* the complexity of ASD can be better explained, whilst also pointing out issues that should be avoided by all means.

5. Discussion

5.1. Contribution to knowledge on the dark side of ASD methodologies

To discuss the core contribution of our work, we derive five propositions that can inform academia and practice about the most prominent characteristics of the dark side of ASD. These propositions aim to address *what* aspects specifically characterize the undesirable side of ASD methodologies and *how* its broad spectrum comes into existence.

Proposition 1. ***Excessive time and delivery pressure entail negative consequences for developer well-being and product quality, thus calling for a sustainable pace.***

Tight deadlines and *delivery pressure for developers* are among the most frequently reported issues in the context of ASD methodologies. Our findings suggests that environments where ASD is practiced can resemble a pressure cooker for developers, making it difficult to get a break from delivering new code within short time frames. *Tight deadlines* in combination with accompanying *iterative delivery pressure for developers* [S4,S29,S46,S52,S59] seem particularly pressing, as these two issues can imply detrimental consequences for the developers and the product quality. Regarding the consequences for developers, *increased workloads* [S37,S38,S53,S68] and excessive *stress* [S4,S5,S11,S29,S46, S52] can result. The frequently reported *overhead* caused by meetings similarly seems to imply further negative effects on developer well-being, i.e., *resource-depletion* and *stress* [S46,S65]. In general, *stress* caused by different forms of *pressure* in ASD was the most frequently reported relationship in our analysis, implying harmful consequences for the physical and psychological health of the development staff.

Simultaneously, *product quality* can drastically decline in high-pressure ASD environments. Especially *short deadlines* seem to force developers into using shortcuts that accumulate *technical debt* [S18,S42, S57]. Handling deficiencies becomes similarly difficult due to *tight schedules*, as no time remains for required *refactoring*, which is therefore frequently *neglected* [S40,S42]. This *negligence of refactoring* further reinforces the accumulation of *technical debt* [S12,S42,S53,S61], which reduces the *overall quality and performance* of the *software* [S10,S18, S25]. Similarly, taking shortcuts regarding the documentation of the development work to meet a deadline was found to be a consequence of the pressurizing nature of ASD [S10,S52,S61,S64].

Altogether, we suggest that the pressure on developers to deliver new software within tight schedules represents the largest threat to developer well-being and long-term product quality. Especially the increased stress levels that can affect developer health, with consequences that include burnout and resource depletion, are important to consider. That said, also the continuous accumulation of technical debt that causes a decline in product quality requires serious attention. We therefore propose that protecting teams from experiencing extensive iteration pressure due to exceedingly tight deadlines and unbalanced workloads is of essential importance. Ensuring a sustainable pace in development is therefore a key task for the scrum master and the product owner. Failing to do so may imply severe consequences for the well-being of developers, which can ultimately lead to burnout. Since excessive delivery pressure also strongly relates to declines in product quality, overwhelming the team with too much work per iteration should also be avoided to ensure consistent delivery of high-quality software.

Proposition 2. ***Ensuring customer involvement and minimizing meeting overhead is paramount for the productivity of ASD teams and the ability to deliver efficiently.***

Customer misbehavior and *meeting overhead* caused by the various ASD ceremonies can imply problematic consequences for the progress of the team. *Inadequate involvement, responsiveness, and availability of the client, lacking customer feedback,* and *requirements volatility* were among the most frequently described issues regarding customer behavior, causing *productivity losses, RE management issues, delivery delays,* and *communication lapses*. Additionally, *lacking customer involvement and availability* seems to precede more issues than any other issue in other study, see Fig. 12 and Table 12, emphasizing the importance of adequate customer participation in ASD. In this vein, especially the *productivity* of the *development team* is affected by inadequate customer behavior [S18,S59, S65,S67], as difficulties in "getting requirements from the business stakeholders [may result] in unnecessary delays and loss of productivity" [S67, p. 2164]. An *unresponsive customer* was furthermore found to force developers into making *inadequate assumptions* about requirements, which may result in products that do not satisfy the customers' needs [S1,S18,S27,S67]. Further potential consequences that underline the effects of *inadequate customer behavior* include *required rework* [S18,S27,S67] that *reduces the progress* and *higher costs* that stem from *required amendments* [S23,S27,S61,S67] to address incorrectly implemented requirements. *Productivity losses* in general represent the most frequently reported issue in our study, see Table 7 in Section 4.4. Decreases in *productivity* were similarly identified as the most frequently affected issue by other antecedent problems, as shown in Fig. 12. Besides *inadequate customer access and involvement* [S18,S59,S66,S67], the extensive *meeting overhead* [S46,S50,S53,S64] resulting from various ceremonies in ASD was found as particularly influential. Especially the frequency of meetings in ASD is often perceived as disruptive to the workflow [S53,S67,S70], implying further threats to *productivity*.

Since team productivity is critical to ensure on-time delivery and frequent provision of software, we propose that shielding ASD teams from potential threats to productivity is a fundamental objective in ASD projects. In this regard, continuous customer involvement and frequent feedback cycles with clients to validate requirements are crucial for highly productive development teams to ensure frequent and timely deliveries. Inadequate requirements assumptions causing deviations from the actual demands that require costly rework and consequently imply schedule deviations can thereby also be avoided. To add, finding a balanced ratio of meetings to avoid excessive overhead and distracting disruptions caused by these ceremonies appears critical for SD teams to maximize the time spent on development tasks. In sum, sustaining the productivity advantages provided by ASD methodologies only seems feasible when these potential threats are sufficiently avoided.

Proposition 3. ***The fast-paced nature of ASD methodologies can provide benefits in early project stages but may prove problematic as development progresses.***

Considering our findings, the demands of the agile manifesto for "early and continuous delivery of valuable software" in a short-time frequency, where "working software is the primary measure of progress" with a focus on "technical excellence and good design" (Beck et al., 2001), appear conflicting. While ASD can already provide software at early project stages (Overhage and Schlauderer, 2012), several product quality-related issues identified in our analysis emphasize that aligning short-term delivery, technical excellence and working software is difficult. Above all, *technical debt, inadequate architectures, failing to meet requirements* and *required rework* that entails further *cost increases* indicate a certain discrepancy within the claims stated before.

Most notably, the *iterative delivery and time pressure* [S52,S55] seem to be the major causes for the *negligence of refactoring,* which aggravates the accumulation of *technical debt* [S12,S42,S53,S61] and thus undermines *product quality*. This observation accentuates that delivering high-quality software under pressure is challenging to achieve. Moreover, the long-term consequences of *technical debt* for *product quality* [S18] and the *disregard of non-functional requirements* for the internal *performance of the software* [S15,S25] appear critical. Similarly, the rapid delivery of software in the early phases of a project without designing an architecture that supports future demands entails negative consequences for *product quality* and *performance*. Additional *rework* to address the quality issues ultimately causes significant *cost increases* [S23,S27,S61,S67].

Considering the negative effects ASD may entail for product quality, we argue that providing high-quality software within short deadlines and as early as possible, whilst accepting new requirements at any stage, appears hardly reconcilable. In addition, the reported limited long-term suitability of architectures that were gradually developed without far-sighted planning, suggests that continuously delivering valuable, working software while incorporating new requirements at any point in development, as proclaimed by the agile manifesto (Beck et al., 2001), can be difficult to attain. To sustain the delivery of high-quality software, granting sufficient time to the planning and development of a robust architecture at the beginning of a project is a significant priority. Moreover, regarding the negligence of refactoring to remove technical debt and the incorrect implementation of NFRs due to delivery pressure, we argue that less may be more in ASD projects. By exerting less pressure to continuously deliver new functionality and dedicating more time to internal quality aspects, refactoring, and necessary architecture requirements, we suggest that ASD can keep its promise of delivering high-quality software. Failing to balance swift software delivery and required rework efforts can ultimately prove disadvantageous for ASD projects in the long-term perspective.

Proposition 4. ***Inadequate organizational contexts can nurture problematic issues that may arise in the introduction and application of ASD methodologies.***

Our findings indicate that applying ASD in *distributed and scaled settings* can be problematic while applying the methodologies within a *non-agile environment* also seems difficult. *Distributed settings* seem to especially cause *inefficient communication* [S15,S54,S60] and may entail a *reduced interaction among team members* [S15,S54]. Similar observations in regard to communication issues in distributed settings were also reported by Alzoubi et al. (2016), thus corroborating these findings. ASD furthermore seems to require a substantial degree of support within organizations that apply the methodologies. As such, without an *agile-oriented culture, changed mindsets,* and *sufficient management support*, organizations may fail to apply ASD methodologies properly, since the methodologies require a transition "from a command and control/mechanistic worldview to a future of autonomous, self-managed agents in a systemic organisation" [S21, p. 10]. Failing to ensure *management support* and *changing the internal culture* can thus make introducing and applying ASD an insurmountable task for some organizations. Dedicated studies focusing on agile transformations, i.e. Dikert et al. (2016), found similar challenges that may impede the proper adoption of ASD methodologies. That said, the identified issues *frequent staff fluctuation*, a *shortage* of *skilled agile developers*, a *lack of trainings,* and an *incorrect introduction of the methodologies* may further aggravate the situation within organizations that apply ASD.

Taken together, we propose that ensuring an adequate foundation in terms of support and adequacy of the environment depicts a fundamentally important task to make ASD methodologies work properly in organizations. While for distributed and scaled settings several concepts have been developed that can alleviate some of these issues, especially the problems that concern the support and endorsement required for ASD need to be addressed accordingly. Should organizations not provide sufficient sponsorship and support, ASD methodologies seem bound to fail with further aggravating consequences.

Proposition 5. ***The complexity of the dark side of ASD is further increased by root cause issues that trigger a chain reaction of further negative consequences.***

Our analysis lastly reveals that some distinct issues may trigger the occurrence of various subsequently arising problems, thus resembling a chain reaction originating from a certain root cause. The literature described several instances where these distinct issues triggered a variety of further successive issues to occur. As an example, [S67] observed that *unresponsive customers* who ignored requests from the team forced them to make *assumptions about requirements*, which in turn caused a *discrepancy* between the *delivered* and the *actual requirements*. Consequently, teams needed to *rework* the deliverables, which then finally implied significant *cost increases*. Following our analysis represented in Fig. 12, a chain of negative consequences stemming from *Inadequate Customer Involvement* could be described as follows:

> *Inadequate Customer Involvement → Inadequate Requirements Assumptions → Difficult Customer Requirements Meeting → Required Rework → Increased Cost*

To add a second example, [S42] found how increased *time pressure* due to *tight*, approaching *deadlines* caused a low-prioritization of required *refactoring,* which then added significant *technical debt* to the general accumulation caused by necessary shortcuts to meet the deadline. In the long-term perspective, *technical debt* also led to declines in *software quality* [S18], suggesting the following issue chain:

> *Tight Deadlines / Time Pressure → Negligence of Refactoring → Technical Debt → Reduced Software Quality / Performance*

Considering these and other apparent issue chains recognizable in Fig. 12, it is essential for ASD practitioners to identify and avoid the root causes that induce other problems to emerge. By following the paths in Fig. 12 for the most influential and frequent issues, practitioners can recognize what core problems need to be avoided by all means to prevent a chain reaction of further negative consequences. In so doing, professionals could significantly improve the application of ASD in practice. The observed stimulation of subsequent issues can meanwhile provide a new approach for scholars to explain the complexity of the dark side of ASD, as is also addressed in the following section.

5.2. Implications for academia and future research opportunities

Our findings have several implications for scholars who investigate the outcomes of applying ASD methodologies. First, our work demonstrates *what* negative aspects may be associated with ASD along the 90 identified issues. With these results, we address an important research gap that thus far appeared somewhat neglected in extant literature. As such, our work significantly extends the findings by Fitriani et al. (2016) by providing an updated, in-depth analysis of the various issues that incorporates recently emerging insights from the literature. Given that ASD research has especially gained momentum in SE since 2016 (Baham and Hirschheim, 2022) and also in other research domains represented in our analysis, we extend our understanding of *what* negative implications ASD may entail by contributing an updated SLR to the knowledge base. Two decades after the emergence of the methodologies, this depicts a core contribution of our work.

Generally, focusing on an unrestricted spectrum of potential issues beyond a limited scope, i.e., challenges in RE (Inayat et al., 2015) or communication challenges in global SD (Alzoubi et al., 2016), demonstrates that exhaustively answering the question *what* negative aspects ASD can entail requires a multidimensional approach. Our work can provide a significant step in this regard, as it classifies the identified issues to the primary level of manifestation and further elaborates on these negative aspects through coherent issue themes to uncover the breadth of the dark side of ASD. With our work, we thereby also complement extant research focusing on *what* beneficial consequences ASD

methodology use can imply, since the business value of ASD appears significantly more regarded by extant research, for instance in Racheva et al. (2010) and Meckenstock et al. (2022). Akin to their findings, the dark side of ASD also embodies a multidimensional concept, which comprises a multifaceted spectrum of negative characteristics with several manifestation levels involved. By combining prior findings on the ASD business value concept with our systematization, a comprehensive perspective on both positive aspects and potential issues of ASD methodology use can be now established by future research efforts, thereby addressing the question of *what* consequences ASD can entail in general.

In addition, the analysis of relationships among issues can provide a starting point for future investigations into *how* ASD methodologies lead to distinct outcomes. We argue that examining the interplay of different issues and *how* they may either reinforce one another or cause other problems to arise can depict a promising approach toward understanding the underlying mechanisms of ASD methodology effect creation. Potential inquiries could focus on the most frequently described relationships in Section 4.10 and develop explanatory approaches to better understand *how* these identified chain reactions of different issues occur. In this regard, first analogous insights have been proposed by research on the positive consequences of ASD, which investigated stimulating relationships between beneficial effects. As such, Kude et al. (2019) found a positive effect of pair programming on backup behavior, which in turn reinforces team performance. To add, Alami and Krancher (2022) illustrate how Scrum can entail various beneficial effects, i.e., providing psychological safety, increased transparency, better collaboration or improved knowledge sharing, which collectively contribute to improved software quality. These investigations on the positive effects of ASD usage combined with insights from our systematic analysis may thus guide future research that examines *how* the negative implications in particular result from ASD methodology use.

Our work can furthermore serve as an impulse for future work to investigate *when* different outcomes result, i.e., by conducting analyses of the contextual conditions under which benefits can result to a certain degree, before negative consequences begin to set in. Indications from the literature that underline this apparent two-sided nature of the respective outcomes of ASD being subject to different contextual conditions include, for instance, the degree of customer collaboration [S27]. Under inadequate customer cooperation, requirements and feedback appear difficult to attain, which can hamper requirements fulfillment and can lead to delivery delays [S27]. In contrast, a more cooperative customer helps to steer the development and ensures that the team meets the needs of the customer on time (Schlauderer and Overhage, 2013). Still, our analysis also reveals that a very cooperative customer that provides frequent feedback and demands changes can in turn increase requirement volatility and trigger the introduction of new requirements during sprints [S28], which represent major drivers of delivery delays [S8,S23,S28]. Consequently, lacking customer cooperation, yet also interference by the customers can cause several issues. Therefore, identifying a sweet-spot of customer involvement to rule out potentially arising issues appears essential.

Another indication in this regard is observable in [S53]. While practices such as daily meetings can imply beneficial effects such as the exchange of information, *when* practiced too much, especially under increased workloads, negative subjective feelings, and resource depletion may arise [S53]. In addition, iterations with extremely short duration and high workloads can create extensive iteration pressure with negative consequences for the developers and their delivery capability [S29]. Nonetheless, the general time-boxed character of ASD that creates some degree of delivery pressure "is necessary to motivate teams to deliver their goals" [S29, p. 628]. In consideration of this two-sided nature of using ASD, given the varying outcomes under different contextual conditions, hypothesizing the use of ASD and its effects to follow a u-shaped relationship may depict another approach for future research toward understanding *when* ASD delivers certain outcomes, i.e., desirable benefits or negative consequences.

Our analysis indicates, however, that especially those contextual factors, which seem to determine *when* benefits or problems result, are rarely investigated in greater detail. Only a quarter of the studies provided an in-depth characterization of the respective context, with even fewer cases deeply reflecting on the impact of those contextual factors on the resulting outcomes. We therefore call for greater attention to contextual factors in future research efforts to develop an understanding *when* ASD methodologies deliver outcomes, i.e., whether these factors stimulate benefit generation or cause issues to occur.

The last implication concerns the observed lack of confirmatory studies in the literature that would help to develop explanatory approaches for *"if, how, why* and *when"* (Baham and Hirschheim, 2022, p. 5) ASD creates outcomes, especially regarding the negative consequences. While quantitative studies seem generally scarce considering the plethora of qualitative case studies in our sample, especially those studies that employ a confirmatory approach to explain *how* ASD methodologies create negative outcomes seem even rarer. We thus call for more affirmatory and explanatory studies in analogy to [S11,S38, S53] in future works. Following this call would significantly advance our understanding of the underlying effect mechanisms that contribute to the, in some instances, significantly differing outcomes of ASD methodology application.

Taken together, we believe that by addressing *what* negative consequences can arise in ASD and *how* these issues come about, a prerequisite for future research that investigates *"if, how, why* and *when"* (Baham and Hirschheim, 2022, p. 5) ASD creates outcomes is now addressed with our work, especially regarding the associated downsides. Future work can adopt the presented findings in generating explanatory approaches for the thus far less understood underlying effect-creation mechanisms that either generate beneficial implications or cause problems to arise.

5.3. Implications for practical application of ASD methodologies

For practitioners that apply ASD methodologies, our work provides the following implications. Most notably, the systematic overview of potential issues can achieve a more balanced perspective on the effects that ASD methodology use may entail. We demonstrate that ASD can imply challenging problems that contrast the achievable business value of ASD (Meckenstock et al., 2022). Practitioners can benefit from this systematization when trying to identify areas that could prove difficult to maximize the potential of using ASD. That said, the benefits attainable through ASD methodologies can swiftly be replaced by negative consequences, should the necessary foundations and conditions required to properly support the methodologies in practice not be ensured accordingly. In this regard, our findings also highlight that some areas may be particularly susceptible to problems, as they were more frequently described in the literature than other issues. Prominent examples include increased stress levels for developers, reduced productivity, delayed product delivery, lower product quality, and communication issues.

This observation also shows that the breadth of outcomes of ASD methodologies use entails more diverse negative implications than is effectively reported in practical literature. Since the rather beneficial implications often appear at the forefront of industry reports, our systematization can serve as an eye opener for practitioners to draw their attention to issues that they may not be aware of. In consideration of our findings, especially the "Annual State of Agile" reports (VersionOne, 2018, 2019, 2020, 2021) appear somewhat misleading, as only a fraction of the most frequently identified issues in our analysis are similarly addressed in these reports. To add, the recent reports only elaborate on a limited set of challenges that organizations may encounter in the adoption and scaling of ASD, yet the actual negative consequences encountered in daily use are barely touched upon. To better represent the challenging nature of ASD methodologies, especially for the large proportion of yet immature practitioners (VersionOne, 2018, 2019,

2020, 2021), however, these reports should address a wider spectrum of critical issues. Our work can serve as a means to uncover this variety of problems that ASD methodologies may entail, as the complexity of the dark side of ASD becomes apparent in our analysis.

Furthermore, several antecedent issues seem to reinforce the occurrence of various other problems, as described in Section 4.10. This should alert practitioners that the occurrence of certain issues may entail the build-up of several other problems, thereby causing chain reactions with further aggravating consequences. To inform practitioners about these relationships, our findings point towards several driving forces of the dark side of ASD, which require dedicated attention to avoid further negative implications. Especially inadequate customer behavior, i.e., lacking involvement or unavailability of client representatives as well as requirements volatility seem to represent pitfalls that can particularly diminish the productivity of the team. Moreover, short deadlines and constant delivery pressure together with too many meetings seem to imply adverse effects on developer well-being, as they were frequently reported to cause stress and resource depletion. Given these observations in the literature, practitioners must ensure customer involvement and a sustainable pace for developers to avoid further repercussions. Our results highlight various other relationships among issues, which can help practitioners identify further root causes for the issues they may encounter in daily SD practice.

In sum, we hope that our findings reduce the impression that ASD methodologies can be a 'silver bullet' for SD, given the spectrum of undesirable consequences identified in our analysis, which stand in stark contrast to the primarily reported benefits in practitioner literature and previous systematizations on the business value of ASD.

5.4. Limitations

Our findings have limitations that need to be considered. We discuss potential *threats to validity* along *reliability* and *credibility, internal* and *external validity*, as well as *construct validity* (Petersen and Gencel, 2013).

As regards *reliability* and *credibility* concerns, a limited probability of bias cannot be ruled out in any SLR (Kitchenham, 2004). This aspect represents the main limitation of our work, as the literature search and analysis were conducted by a single author. To avoid bias, a senior researcher with extensive experience in SLRs and research on ASD methodologies supervised the author in the literature search and analysis process. The author also thoroughly discussed questionable articles with research colleagues in the selection process. Still, a possibility of bias that may influence the findings remains.

Furthermore, a limited potential for unintentional omission of articles remains in any SLR, potentially influencing both the *reliability* and *credibility* of our research. To avoid omission and include as many relevant articles as possible in the initial review sample, we defined a comprehensive search string that includes 22 different notions related to the negative implications of ASD use. We also employed strict inclusion and exclusion criteria in the selection of relevant articles to further ensure objectivity. In this vein, we need to acknowledge another *reliability*-related limitation that may result from the notions that are included in the search string, since these notions only focused on negative aspects due to the scope of the paper. We can therefore not guarantee that articles primarily investigating positive implications of ASD methodologies, yet also pointing out negative consequences, are equally included in the review sample. Examples include [S11], which investigates "the benefits and burdens of agile ISD practices use for developer well-being" (p. 557), thereby both thematizing the up- and downsides of ASD. Still, we argue that the strong thematic overlap of the analyzed papers and the substantial number of identified issues suggest a comprehensive coverage of relevant literature.

To add, potential threats to *internal validity* and *construct validity*, both pertaining to the *theoretical validity* of our work, also need to be addressed, given the single-author nature of this article. We acknowledge that performing the analysis and coding work by one single coder may cause inconsistencies in the results. To prevent potential inconsistencies, we followed an iterative process to consolidate the identified issues, which was simultaneously monitored by an external senior researcher. Involving this expert helped in further reducing the chance for an inconsistent representation of the facets of the examined topic, as they provided feedback on the analysis results. We therefore argue that our analysis results depict a consistent representation of the dark side of ASD methodologies, albeit other researchers may have come to different interpretations of the identified issues that relate to the focal topic.

Lastly, we discuss *external validity* concerns, since this paper only answers *what* negative outcomes ASD methodology use may imply, yet leaves contextual factors out of scope. As such, the question *when*, i.e., under which circumstances ASD can lead to undesirable effects, remains unanswered by our research. Consequently, it may be difficult to generalize our findings on negative outcomes without taking the respective context factors into account. Depending on the contextual factors, different negative consequences may be more or less likely to appear. While our work does not intend to answer *when* issues occur, our findings can make practitioners more aware of the general negative consequences they may experience, even if they are less likely in their respective cases. To add, as identified, contextual factors seem mostly neglected in the analyzed studies, thus inspecting their effects represents an important avenue for future research. Despite these limitations, we believe our work can provide an important step for future research that investigates "*if, how, why* and *when* ASD impacts outcomes" (Baham and Hirschheim, 2022, p. 5), with a particular focus on the negative aspects of ASD usage.

6. Concluding remarks

Despite the widespread adoption of ASD methodologies in practice and significant advancements made by research to understand the underlying effect mechanisms, the negative implications of ASD thus far remained less comprehensively reflected upon by extant literature. Our work provides a notable step in this regard, as it uncovers the broad spectrum of issues that can occur in ASD methodologies and systematically organizes these problems. The conducted investigation identified 90 different issues and aggregated them into 18 comprehensive issue themes, which were assigned to six levels where these problems primarily manifest themselves. Our findings thereby help to address the question *what* characterizes the dark side of ASD methodologies. Our work also shows that ASD can be a double-edged sword, since the achievable ASD business value seems to stand in direct contrast to the aspects of the dark side that were identified in our analysis. To add, we highlight that these negative consequences can be caused by further occurring issues, thereby underlining the complexity of the underlying effect mechanisms of ASD methodologies. In this regard, the analysis of the relationships between issues can provide significant insights into *how* ASD causes these less desirable effects.

Research and practice may draw several conclusions from our findings. For academia, our work provides a starting point for future research efforts that intend to investigate *how* the wide spectrum of issues in ASD methodology use arises, as some aspects appear better understood, while for some issues a lack of comprehensive insights seems present. In this vein, the identified relationships among issues can depict a potential approach for researchers to explain the mechanisms that lead to these negative outcomes in ASD. To add, integrating our findings with extant classifications on the beneficial aspects that embody the business value of ASD (Meckenstock et al., 2022) can enable a more balanced delineation of the overall value-creation potential of the methodologies. Doing so would furthermore help to answer the question of *what* outcomes ASD can create in general. Researchers may also build on this systematization when developing explanations of *"if, how, why* and *when"* (Baham and Hirschheim, 2022, p. 5) ASD creates specific outcomes, as these may imply concurrent positive and negative aspects.

As for practical implications, our findings inform practitioners about

the most frequent and detrimental aspects of ASD usage. Our analysis also highlights possible root causes of various issues that need to be addressed to maximize the value-creation potential of the methodologies. Practitioners can furthermore identify areas for improvement based on the provided systematization. Lastly, we believe that our work can create more awareness of the downsides of ASD for practitioners who already employ ASD methodologies but are not mature in their application. Similarly, new adopters who intend to introduce them in their organization can benefit from our findings, as the dark side of ASD often seems rather neglected in industry reports. With the stark contrast presented by our findings, we hope that an impression of ASD as a 'silver bullet' for SD can be avoided in both practice and academia.

CRediT authorship contribution statement

Jan-Niklas Meckenstock: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.jss.2024.111966.

Appendix A. Additional information

We provide additional information via an online appendix, which can be accessed under the following link: https://figshare.com/s/20c1a6307d761c08e2a3

Appendix B. Literature study sample

ID	Study in the Literature Sample
[S1]	Alahyari, H., Gorschek, T., & Berntsson Svensson, R. (2019). An exploratory study of waste in software development organizations using agile or lean approaches: A multiple case study at 14 organizations. *Information and Software Technology, 105*, 78–94. https://doi.org/10.1016/j.infsof.2018.08.006
[S2]	Alsaqaf, W., Daneva, M., & Wieringa, R. (2018). Understanding Challenging Situations in Agile Quality Requirements Engineering and Their Solution strategies: Insights from a Case Study. 2018 IEEE 26th International Requirements Engineering Conference, Banff, AB, Canada.
[S3]	Annosi, M. C., Mattarelli, E., Micelotta, E., & Martini, A. (2022). Logics' shift and depletion of innovation: A multi-level study of agile use in a multinational telco company. *Information and Organization, 32*(3), 100,421. https://doi.org/10.1016/j.infoandorg.2022.100421
[S4]	Annosi, M. C., Magnusson, M., Martini, A., & Appio, F. P. (2016). Social Conduct, Learning and Innovation: An Abductive Study of the Dark Side of Agile Software Development. *Creativity and Innovation Management, 25*(4), 515–535. https://doi.org/10.1111/caim.12172
[S5]	Annosi, M. C., Foss, N., & Martini, A. (2020). When Agile Harms Learning and Innovation: (and What Can Be Done About It). *California Management Review, 63*(1), 61–80. https://doi.org/10.1177/0008125620948265
[S6]	Asnawi, A. L., Gravell, A. M., & Wills, G. B. (2011). Empirical Investigation on Agile Methods Usage: Issues Identified from Early Adopters in Malaysia. XP 2011, Madrid, Spain.
[S7]	Babar, M. A. (2009). An exploratory study of architectural practices and challenges in using agile software development approaches. 2009 Joint Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture, Cambridge, UK.
[S8]	Badampudi, D., Fricker, S. A., & Moreno, A. M. (2013). Perspectives on Productivity and Delays in Large-Scale Agile Projects. XP 2013, Vienna, Austria.
[S9]	Basten, D., Stavrou, G., & Pankratz, O. (2016). Closing the Stakeholder Expectation Gap: Managing Customer Expectations toward the Process of Developing Information Systems. *Project Management Journal, 47*(5), 70–88. https://doi.org/10.1177/875697281604700506
[S10]	Behutiye, W., Rodríguez, P., Oivo, M., Aaramaa, S., Partanen, J., & Abhervé, A. (2022). Towards optimal quality requirement documentation in agile software development: A multiple case study. *Journal of Systems and Software, 183*, 111,112. https://doi.org/10.1016/j.jss.2021.111112
[S11]	Benlian, A. (2022). Sprint Zeal or Sprint Fatigue? The Benefits and Burdens of Agile ISD Practices Use for Developer Well-Being. *Information Systems Research, 33*(2), 557–578. https://doi.org/10.1287/isre.2021.1069
[S12]	Codabux, Z., & Williams, B. (2013). Managing technical debt: An industrial case study. 2013 4th International Workshop on Managing Technical Debt (MTD), San Francisco, CA, USA.
[S13]	Conboy, K., Coyle, S., Wang, X., & Pikkarainen, M. (2011). People over Process: Key Challenges in Agile Development. *IEEE Software, 28*(4), 48–57. https://doi.org/10.1109/ms.2010.132
[S14]	Coyle, S., Conboy, K., & Acton, T. (2015). An Exploration of the relationship between Contribution Behaviours and the Decision Making Process in Agile Teams. ICIS 2015, Fort Worth, TX, USA.
[S15]	Dasanayake, S., Aaramaa, S., Markkula, J., & Oivo, M. (2019). Impact of requirements volatility on software architecture: How do software teams keep up with ever-changing requirements? *Journal of Software: Evolution and Process, 31*(6). https://doi.org/10.1002/smr.2160
[S16]	de O. Melo, C., S. Cruzes, D., Kon, F., & Conradi, R. (2013). Interpretative case studies on agile team productivity and management. *Information and Software Technology, 55*(2), 412–427. https://doi.org/10.1016/j.infsof.2012.09.004
[S17]	Drury, M., Conboy, K., & Power, K. (2012). Obstacles to decision making in Agile software development teams. *Journal of Systems and Software, 85*(6), 1239–1254. https://doi.org/10.1016/j.jss.2012.01.058
[S18]	Elbanna, A. R. (2014). Identifying the Risks associated with Agile Software Development: an Empirical Investigation. Proceedings of the 8th Mediterranean Conference on Information Systems, Verona, Italy.
[S19]	Flouri, K., & Berger, H. (2010). Agile Development: Agile Adopted in Practice but not in Principle. UK Academy for Information Systems Conference Proceedings 2010, Oxford, UK.
[S20]	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
[S21]	Gregory, P., Barroca, L., Sharp, H., Deshpande, A., & Taylor, K. (2016). The challenges that challenge: Engaging with agile practitioners' concerns. *Information and Software Technology, 77*, 92–104. https://doi.org/10.1016/j.infsof.2016.04.006

(*continued on next page*)

(*continued*)

ID	Study in the Literature Sample
[S22]	Gregory, P., Barroca, L., Taylor, K., Salah, D., & Sharp, H. (2015). Agile Challenges in Practice: A Thematic Analysis. XP 2015, Helsinki, Finland.
[S23]	Hannay, J. E., & Benestad, H. C. (2010). Perceived productivity threats in large agile development projects. Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, New York, NY, USA.
[S24]	Heikkilä, V. T., Paasivaara, M., Rautiainen, K., Lassenius, C., Toivola, T., & Järvinen, J. (2015). Operational release planning in large-scale Scrum with multiple stakeholders – A longitudinal case study at F-Secure Corporation. *Information and Software Technology, 57*, 116–140. https://doi.org/10.1016/j.infsof.2014.09.005
[S25]	Heikkilä, V. T., Paasivaara, M., Lasssenius, C., Damian, D., & Engblom, C. (2017). Managing the requirements flow from strategy to release in large-scale agile development: a case study at Ericsson. *Empirical Software Engineering, 22*(6), 2892–2936. https://doi.org/10.1007/s10664–016–9491-z
[S26]	Hess, A., Diebold, P., & Seyff, N. (2019). Understanding information needs of agile teams to improve requirements communication. *Journal of Industrial Information Integration, 14*, 3–15. https://doi.org/10.1016/j.jii.2018.04.002
[S27]	Hoda, R., Noble, J., & Marshall, S. (2011a). The impact of inadequate customer collaboration on self-organizing Agile teams. *Information and Software Technology, 53*(5), 521–534. https://doi.org/10.1016/j.infsof.2010.10.009
[S28]	Hoda, R., & Murugesan, L. K. (2016). Multi-level agile project management challenges: A self-organizing team perspective. *Journal of Systems and Software, 117*, 245–257. https://doi.org/10.1016/j.jss.2016.02.049
[S29]	Hoda, R., Noble, J., & Marshall, S. (2011b). Developing a grounded theory to explain the practices of self-organizing Agile teams. *Empirical Software Engineering, 17*(6), 609–639. https://doi.org/10.1007/s10664–011–9161–0
[S30]	Holvitie, J., Licorish, S. A., Spínola, R. O., Hyrynsalmi, S., MacDonell, S. G., Mendes, T. S., Buchan, J., & Leppänen, V. (2018). Technical debt and agile software development practices and processes: An industry practitioner survey. *Information and Software Technology, 96*, 141–160. https://doi.org/10.1016/j.infsof.2017.11.015
[S31]	Kalenda, M., Hyna, P., & Rossi, B. (2018). Scaling agile in large organizations: Practices, challenges, and success factors. *Journal of Software: Evolution and Process, 30*(10). https://doi.org/10.1002/smr.1954
[S32]	Kalinowski, M., Felderer, M., Conte, T., Spínola, R., Prikladnicki, R., Winkler, D., Fernández, D. M., & Wagner, S. (2016). Preventing Incomplete/Hidden Requirements: Reflections on Survey Data from Austria and Brazil. SQWD 2016: Software Quality. The Future of Systems- and Software Development, Vienna, Austria.
[S33]	Kamei, F., Pinto, G., Cartaxo, B., & Vasconcelos, A. (2017). On the Benefits/Limitations of Agile Software Development. Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering, Karlskrona, Sweden.
[S34]	Katumba, B., & Knauss, E. (2014). Agile Development in Automotive Software Development: Challenges and Opportunities. PROFES 2014: Product-Focused Software Process Improvement, Helsinki, Finland.
[S35]	Khanagha, S., Volberda, H. W., Alexiou, A., & Annosi, M. C. (2021). Mitigating the dark side of agile teams: Peer pressure, leaders' control, and the innovative output of agile teams. *Journal of Product Innovation Management, 39*(3), 334–350. https://doi.org/10.1111/jpim.12589
[S36]	Kopczyńska, S., Ochodek, M., Piechowiak, J., & Nawrocki, J. (2022). On the benefits and problems related to using Definition of Done — A survey study. *Journal of Systems and Software, 193*, 111,479. https://doi.org/10.1016/j.jss.2022.111479
[S37]	Laanti, M. (2013). Agile and Wellbeing – Stress, Empowerment, and Performance in Scrum and Kanban Teams. 2013 46th Hawaii International Conference on System Sciences, Hawaii, HI, USA.
[S38]	Laanti, M., Salo, O., & Abrahamsson, P. (2011). Agile methods rapidly replacing traditional methods at Nokia: A survey of opinions on agile transformation. *Information and Software Technology, 53*(3), 276–290. https://doi.org/10.1016/j.infsof.2010.11.010
[S39]	Lagerberg, L., Skude, T., Emanuelsson, P., Sandahl, K., & Stahl, D. (2013). The Impact of Agile Principles and Practices on Large-Scale Software Development Projects: A Multiple-Case Study of Two Projects at Ericsson. 2013 ACM / IEEE International Symposium on Empirical Software Engineering and Measurement, Baltimore, MD, USA.
[S40]	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
[S41]	Marchenko, A., & Abrahamsson, P. (2008). Scrum in a Multiproject Environment: An Ethnographically-Inspired Case Study on the Adoption Challenges. Agile 2008 Conference, Toronto, ON, Canada.
[S42]	Martini, A., Bosch, J., & Chaudron, M. (2015). Investigating Architectural Technical Debt accumulation and refactoring over time: A multiple-case study. *Information and Software Technology, 67*, 237–253. https://doi.org/10.1016/j.infsof.2015.07.005
[S43]	Matthies, C., & Dobrigkeit, F. (2020). Towards Empirically Validated Remedies for Scrum Retrospective Headaches. Proceedings of the 53rd Hawaii International Conference on System Sciences, Hawaii, HI, USA.
[S44]	Matthies, C., Dobrigkeit, F., & Ernst, A. (2019). Counteracting Agile Retrospective Problems with Retrospective Activities. EuroSPI 2019: Systems, Software and Services Process Improvement, Edinburgh, UK.
[S45]	McAvoy, J., & Butler, T. (2017). The role of project management in ineffective decision making within Agile software development projects. *European Journal of Information Systems, 18*(4), 372–383. https://doi.org/10.1057/ejis.2009.22
[S46]	McHugh, O., Conboy, K., & Lang, M. (2011). Using agile practices to build trust in an agile team: A case study. Proceedings of International Conference on Information Systems Development (ISD2010), Prague, Czech Republic.
[S47]	Meier, A., Kropp, M., Anslow, C., & Biddle, R. (2018). Stress in Agile Software Development: Practices and Outcomes. XP 2018, Porto, Portugal.
[S48]	Moe, N. B., & Aurum, A. (2008). Understanding Decision-Making in Agile Software Development: A Case-study. 2008 34th Euromicro Conference Software Engineering and Advanced Applications, Parma, Italy.
[S49]	Moe, N. B., Dahl, B., Stray, V., Karlsen, L. S., & Schjødt-Osmo, S. (2019). Team Autonomy in Large-Scale Agile. Proceedings of the 52nd Hawaii International Conference on System Sciences, Hawaii, HI, USA.
[S50]	Moe, N. B., Dingsøyr, T., & Dybå, T. (2010). A teamwork model for understanding an agile team: A case study of a Scrum project. *Information and Software Technology, 52*(5), 480–491. https://doi.org/10.1016/j.infsof.2009.11.004
[S51]	Moe, N. B., Aurum, A., & Dybå, T. (2012). Challenges of shared decision-making: A multiple case study of agile software development. *Information and Software Technology, 54*(8), 853–865. https://doi.org/10.1016/j.infsof.2011.11.006
[S52]	Mohallel, A. A., & Bass, J. M. (2019). Agile Software Development Practices in Egypt SMEs: A Grounded Theory Investigation. ICT4D 2019, Dar es Salaam, Tanzania.
[S53]	Mueller, L., & Benlian, A. (2022). Too Drained from Being Agile? The Self-Regulatory Effects of the Use of Agile ISD Practices and their Consequences on Turnover Intention. *Journal of the Association for Information Systems, 23*(6), 1420–1455. https://doi.org/10.17705/1jais.00766
[S54]	Nuottila, J., Aaltonen, K., & Kujala, J. (2016). Challenges of adopting agile methods in a public organization. *International Journal of Information Systems and Project Management, 4*(3), 65–85. https://doi.org/10.12821/ijispm040304
[S55]	O'Donnell, M. J., & Richardson, I. (2008). Problems Encountered When Implementing Agile Methods in a Very Small Company. EuroSPI 2008: Software Process Improvement, Dublin, Ireland.
[S56]	Olszewska, M., Heidenberg, J., Weijola, M., Mikkonen, K., & Porres, I. (2016). Quantitatively measuring a large-scale agile transformation. *Journal of Systems and Software, 117*, 258–273. https://doi.org/10.1016/j.jss.2016.03.029
[S57]	Paasivaara, M., Behm, B., Lassenius, C., & Hallikainen, M. (2018). Large-scale agile transformation at Ericsson: a case study. *Empirical Software Engineering, 23*(5), 2550–2596. https://doi.org/10.1007/s10664–017–9555–8
[S58]	Petersen, K., & Wohlin, C. (2009). A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case. *Journal of Systems and Software, 82*(9), 1479–1490. https://doi.org/10.1016/j.jss.2009.03.036
[S59]	Rahy, S., & Bass, J. (2020). Overcoming Team Boundaries in Agile Software Development. *Journal of International Technology and Information Management, 29*(4), 20–49. https://doi.org/10.58729/1941–6679.1433
[S60]	Ralph, P., & Shportun, P. (2013). Scrum Abandonment in Distributed Teams: A Revelatory Case. PACIS 2013 Proceedings, Jeju Island, Korea.
[S61]	Ramesh, B., Cao, L., & Baskerville, R. (2007). 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

(*continued on next page*)

(*continued*)

ID	Study in the Literature Sample
[S62]	Rodríguez, P., Markkula, J., Oivo, M., & Turula, K. (2012). Survey on agile and lean usage in finnish software industry. Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement, New York, NY, USA.
[S63]	Schön, E.-M., Winter, D., Escalona, M. J., & Thomaschewski, J. (2017). Key Challenges in Agile Requirements Engineering. XP 2017, Cologne, Germany.
[S64]	Singh, K., & Strobel, J. (2022). Exploring lived experiences of agile developers with daily stand-up meetings: a phenomenological study. *Behaviour & Information Technology, 42*(4), 403–423. https://doi.org/10.1080/0144929x.2021.2023636
[S65]	Stray, V., Sjøberg, D. I. K., & Dybå, T. (2016). The daily stand-up meeting: A grounded theory study. *Journal of Systems and Software, 114*, 101–124. https://doi.org/10.1016/j.jss.2016.01.004
[S66]	Stray, V., Moe, N. B., & Dingsøyr, T. (2011). Challenges to Teamwork: A Multiple Case Study of Two Agile Teams. XP 2011, Madrid, Spain.
[S67]	van Waardenburg, G., & van Vliet, H. (2013). When agile meets the enterprise. *Information and Software Technology, 55*(12), 2154–2171. https://doi.org/10.1016/j.infsof.2013.07.012
[S68]	Vanhala, E., & Kasurinen, J. (2019). The Role of the Customer in an Agile Project: A Multi-case Study.*Lecture Notes in Business Information Processing* ICSOB 2019: Software Business, Jyväskylä, Finland.
[S69]	Waterman, M., Noble, J., & Allan, G. (2015). How Much Up-Front? A Grounded theory of Agile Architecture. 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, Florence, Italy.
[S70]	Wiesche, M. (2021). Interruptions in Agile Software Development Teams. *Project Management Journal, 52*(2), 210–222. https://doi.org/10.1177/8756972821991365

References

Abrantes, J.F., Travassos, G.H., 2011. Common agile practices in software processes. In: 2011 International Symposium on Empirical Software Engineering and Measurement. Banff, AB, Canada.

Alami, A., Krancher, O., 2022. How Scrum adds value to achieving software quality? Empir Softw Eng 27 (7), 165. https://doi.org/10.1007/s10664-022-10208-4.

Alzoubi, Y.I., Gill, A.Q., Al-Ani, A., 2016. Empirical studies of geographically distributed agile development communication challenges: a systematic review. Inf. Manage. 53 (1), 22–37. https://doi.org/10.1016/j.im.2015.08.003.

Baham, C., Hirschheim, R., 2022. Issues, challenges, and a proposed theoretical core of agile software development research. Inf. Syst. J. 32 (1), 103–129. https://doi.org/10.1111/isj.12336.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., & others. (2001). Manifesto for agile software development.

Chow, T., Cao, D.-B., 2008. A survey study of critical success factors in agile software projects. J. Syst. Softw. 81 (6), 961–971. https://doi.org/10.1016/j.jss.2007.08.020.

Conboy, K., 2009. Agility from first principles: reconstructing the concept of agility in information systems development. Inf. Syst. Res. 20 (3), 329–354. https://doi.org/10.1287/isre.1090.0236.

Dikert, K., Paasivaara, M., Lassenius, C., 2016. Challenges and success factors for large-scale agile transformations: a systematic literature review. J. Syst. Softw. 119, 87–108. https://doi.org/10.1016/j.jss.2016.06.013.

Dingsøyr, T., Nerur, S., Balijepally, V., Moe, N.B., 2012. A decade of agile methodologies: towards explaining agile software development. J. Syst. Softw. 85 (6), 1213–1221. https://doi.org/10.1016/j.jss.2012.02.033.

Dybå, T., Dingsøyr, T., 2008. Empirical studies of agile software development: a systematic review. Inf. Softw. Technol. 50 (9–10), 833–859. https://doi.org/10.1016/j.infsof.2008.01.006.

Elbanna, A., Murray, D., 2009. Organizing projects for innovation: a collective mindfulness perspective. In: AMCIS 2009 Proceedings. San Francisco, CA, USA.

Fitriani, W.R., Rahayu, P., Sensuse, D.I., 2016. Challenges in agile software development: a systematic literature review. In: 2016 International Conference on Advanced Computer Science and Information Systems (ICACSIS). Malang, Indonesia.

Fitzgerald, B., Hartnett, G., Conboy, K., 2006. Customising agile methods to software practices at Intel Shannon. Eur. J. Inf. Syst. 15 (2), 200–213. https://doi.org/10.1057/palgrave.ejis.3000605.

Gioia, D.A., Corley, K.G., Hamilton, A.L., 2013. Seeking Qualitative Rigor in Inductive Research. Organ. Res. Methods 16 (1), 15–31. https://doi.org/10.1177/1094428112452151.

Hoda, R., Salleh, N., Grundy, J., Tee, H.M., 2017. Systematic literature reviews in agile software development: a tertiary study. Inf. Softw. Technol. 85, 60–70. https://doi.org/10.1016/j.infsof.2017.01.007.

Inayat, I., Salim, S.S., Marczak, S., Daneva, M., Shamshirband, S., 2015. A systematic literature review on agile requirements engineering practices and challenges. Comput. Human Behav. 51, 915–929. https://doi.org/10.1016/j.chb.2014.10.046.

Jalali, S., Wohlin, C., 2011. Global software engineering and agile practices: a systematic review. J. Softw.: Evol. Process 24 (6), 643–659. https://doi.org/10.1002/smr.561.

Kitchenham, B., 2004. Procedures For Performing Systematic Reviews. Keele, UK.

Kitchenham, B., & Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering.

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

Kupiainen, E., Mäntylä, M.V., Itkonen, J., 2015. Using metrics in agile and lean software development – a systematic literature review of industrial studies. Inf. Softw. Technol. 62, 143–163. https://doi.org/10.1016/j.infsof.2015.02.005.

Meckenstock, J.-N., Hirschlein, N., Schlauderer, S., Overhage, S., 2022. The business value of agile software development: results from a systematic literature review. In: ECIS 2022 Proceedings. Timisoara, Romania.

Overhage, S., Schlauderer, S., 2012. How sustainable are agile methodologies? Acceptance factors and developer perceptions in Scrum projects. In: ECIS 2012 Proceedings. Barcelona, Spain.

Overhage, S., Schlauderer, S., Birkmeier, D., Miller, J., 2011. What makes IT personnel adopt scrum? A framework of drivers and inhibitors to developer acceptance. In: Hawaii International Conference on System Sciences 2011 (HICSS-44). Hawaii, HI, USA.

Paré, G., Trudel, M.-C., Jaana, M., Kitsiou, S., 2015. Synthesizing information systems knowledge: a typology of literature reviews. Inf. Manage. 52 (2), 183–199. https://doi.org/10.1016/j.im.2014.08.008.

Paré, G., Wagner, G., Prester, J., 2023. How to develop and frame impactful review articles: key recommendations. J. Decision Syst. 1–17. https://doi.org/10.1080/12460125.2023.2197701.

Petersen, K., Gencel, C., 2013. In: Worldviews, Research Methods, and their Relationship to Validity in Empirical Software Engineering Research. 2013 Joint Conference of the 23rd International Workshop on Software Measurement and the 8th International Conference on Software Process and Product Measurement. Ankara, Turkey.

Racheva, Z., Daneva, M., Sikkel, K., 2009. Value creation by agile projects: methodology or mystery? PROFES 2009: Product-Focused Software Process Improvement Oulu, Finland.

Racheva, Z., Daneva, M., Sikkel, K., Buglione, L., 2010. Business value is not only dollars – results from case study research on agile software projects. PROFES 2010: Product-Focused Software Process Improvement. Limerick, Ireland.

Rohunen, A., Rodriguez, P., Kuvaja, P., Krzanik, L., Markkula, J., 2010. Approaches to Agile Adoption in Large Settings: a Comparison of the Results from a Literature Analysis and an Industrial Inventory. PROFES 2010: Product-Focused Software Process Improvement. Limerick, Ireland.

Sandberg, J., Alvesson, M., 2010. Ways of constructing research questions: gap-spotting or problematization? Organization 18 (1), 23–44. https://doi.org/10.1177/1350508410372151.

Schlauderer, S., Overhage, S., 2013. Exploring the customer perspective of agile development: acceptance factors and on-site customer perceptions in Scrum projects. In: ICIS 2013 Proceedings. Milan, Italy.

Selleri Silva, F., Soares, F.S.F., Peres, A.L., Azevedo, I.M.d., Vasconcelos, A.P.L.F., Kamei, F.K., Meira, S.R.d.L, 2015. Using CMMI together with agile software development: a systematic review. Inf. Softw. Technol. 58, 20–43. https://doi.org/10.1016/j.infsof.2014.09.012.

Strauss, A., Corbin, J., 1990. Basics of Qualitative Research. Sage publications.

Tarhan, A., Yilmaz, S.G., 2014. Systematic analyses and comparison of development performance and product quality of Incremental Process and Agile Process. Inf. Softw. Technol. 56 (5), 477–494. https://doi.org/10.1016/j.infsof.2013.12.002.

Tripp, J., Riemenschneider, C., Thatcher, J., 2016. Job Satisfaction in Agile Development Teams: agile Development as Work Redesign. Journal of the Assoc.r Inf. Syst. 17 (4), 267–307. https://doi.org/10.17705/1jais.00426.

Usman, M., Mendes, E., Weidt, F., Britto, R., 2014. Effort estimation in agile software development: a systematic literature review. In: Proceedings of the 10th International Conference on Predictive Models in Software Engineering. Turin, Italy.

VersionOne. (2018). *11th Annual State of Agile Report*. https://www.agile247.pl/wp-content/uploads/2017/04/versionone-11th-annual-state-of-agile-report.pdf.

VersionOne. (2019). *12th Annual State of Agile Report*. https://www.qagile.pl/wp-content/uploads/2018/04/versionone-12th-annual-state-of-agile-report.pdf.

VersionOne. (2020). *13th Annual State of Agile Report*. https://www.duxdiligens.com/wp-content/uploads/2019/09/13th-annual-state-of-agile-report_7_May_2019.pdf.

VersionOne. (2021). 14th Annual State of Agile Report. https://www.qagile.pl/wp-content/uploads/2020/06/14th-annual-state-of-agile-report.pdf.

Vidgen, R., Wang, X., 2009. Coevolving systems and the organization of agile software development. Inf. Syst. Res. 20 (3), 355–376. https://doi.org/10.1287/isre.1090.0237.

Wohlin, C., 2014. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering. London, England, UK.

Wolfswinkel, J.F., Furtmueller, E., Wilderom, C.P.M, 2017. Using grounded theory as a method for rigorously reviewing literature. Eur. J. Inf. Syst. 22 (1), 45–55. https://doi.org/10.1057/ejis.2011.51.

Jan-Niklas Meckenstock is a Ph.D. student and Research Assistant at the Chair of Industrial Information Systems, University of Bamberg. He holds an M.*Sc*. in International Information Systems Management from the University of Bamberg. His main research interest focuses on agile software development, particularly the effects of ASD methodology application on individuals and the development organization.