

Scrum in practice: an overview of Scrum adaptations

Michal Hron
Aarhus University
michal.hron@post.au.dk

Nikolaus Obwegeser
Aarhus University
nikolaus@mgmt.au.dk

Abstract

Agile software development practices have gained widespread acceptance and application across all industries. Scrum, as one of the most widely used agile methods, has been adopted in countless organizations. However, while there is an understanding that practitioners rarely apply Scrum “by the book”, only little research addresses the actual adaptations and modifications that are made to fit Scrum to real world requirements: whether it is to solve methodological drawbacks, to fit the method to specific contextual constraint, or to add additional value to the method by augmentation or combination with other tools and methods. To get an overview of the proposed adaptations and their implications, this study presents a systematic review of literature reporting on challenges and motivations that lead to modifications of the Scrum method. Based on 31 relevant studies we extract seven distinct motivations for modifying Scrum, as well as six generic solution strategies to adapt the method.

1. Introduction

In the context of software development, agile development methods have been originally conceived with small, co-located teams of software developer generalists in mind. As agile development methods grew in acceptance, they were introduced to a multitude of different settings that depart from the original, idealized picture, and thus the methods had to be adapted to a variety of contexts. In addition, practitioners are continuously raising their expectations to what agile development approaches can deliver. That is in particular with respect to management-related activities such as estimation, reporting, or alignment of software development activities with business strategy.

One of the most popular agile development frameworks is Scrum [45], due to its simplicity and consequent versatility. In a yearly conducted “State of Agile Report” [45], Scrum (and combinations of

Scrum with other techniques) consequently occupies more than half of all agile techniques that are reportedly in use.

In this study, we use Scrum as a window into the agile world, based on its high level of diffusion and practical acceptance. We aim to look for insights on the application of Scrum in practice: what are commonly faced limitations? What are typically suggested alterations of Scrum to those circumstances? Our goal is to get an overview of the motivations as to why one would modify, or add to, the Scrum method, as well as to understand the commonly used solution strategies applied to perform these modifications. Based on our analysis and synthesis of existing modifications we are able to provide a structured overview of the current body of knowledge and propose promising suggestions for future method development.

2. Background

Agile development is a development philosophy standing as a counterpart to traditional, plan-based, “waterfall” approaches [2]. In information systems development (ISD), agility refers to “the continual readiness of an ISD method to rapidly or inherently create change, proactively or reactively embrace change, and learn from change while contributing to perceived customer value (economy, quality, and simplicity), through its collective components and relationships with its environment” [8:340]. The agile approach is attempting to account for the inherent unpredictability of the software development process by taking an incremental approach to development, minimizing planning, estimation, and other overhead tasks, and establishing continuous communication and interaction with the customer. Agile teams continually ship working features in order to maximize impact and reduce time-to-market of new developments. While a plethora of agile ISD methods have been proposed, agile development frameworks and methods are typically not implementable without being tailored to the unique circumstances of the specific development environment [12, 13].

Scrum was first introduced in 1997 [40], and has since become the most widely applied agile software development framework [45]. At its core, Scrum splits development into iterations not longer than four weeks (called sprints). At the end of each sprint, a shippable product increment is delivered to the user. For each new sprint, a sprint-planning meeting is held, at which tasks for the sprint are selected by the developers themselves in collaboration with other stakeholders. In Scrum, the customer is represented in the role of the product owner. Requirements are captured in the form of user stories and are aggregated in a prioritized product backlog. The product backlog is a “living” document, as it is updated continuously and thus reflecting the current understanding of user needs.

In its original form, Scrum is designed for small interdisciplinary teams of about six to nine developers. An important property of any Scrum team is self-organization: i.e., the team itself has the authority to decide on strategies to achieve the objectives of the sprint. To coordinate the daily work and the adherence to the Scrum process, the role of the Scrum master is required in every Scrum team.

Quick pace of work is maintained by daily stand-up meetings, during which team members inform each other about their progress and tasks for the day. Learning is facilitated through so-called retrospectives, which take place after each sprint and provide room for reflection on the work practices of the concluded sprint.

3. Related Work

In this study, we are interested in understanding Scrum in practice—i.e., why and how Scrum was adapted in real-world application. While some previous review studies pursued similar goals, we argue that the underlying research differs from prior work in two main aspects: contextual focus (i.e., limitations to a particular setting) and breadth of methods under investigation (i.e., agile methods in general).

Previous literature reviews typically focus on exploring adopted practices under one specific set of circumstances, e.g., agile in the context of global software development, or they follow one specific motivation, e.g., incorporating user experience design (UX) practices into agile development. As much as they are narrower in the circumstance studied, they are broader in the methodologies in question. They typically look at agile software development globally without limiting themselves to a specific methodology.

In contrast, this literature review presents a map of situations that motivated adjustments of a single method—Scrum. Due to its widespread use and dominant position among agile ISD methods, we focused on adjustments made to Scrum. However, we argue that Scrum may act as a window to the agile development world, and that our findings therefore may well be carefully related to other, similar methods.

Among the previously conducted literature studies, the following stand out: Hossain, Babar, and Paik (2009) and Jalali and Wohlin (2010) have both mapped agile practices in global software engineering. They arrive at similar conclusions and identify comparable practices employed to counter those challenges. Such studies usually take the form of methodology guidance and discussion of best practices, which is consistent with our findings.

Duechting, Zimmermann, and Nebe (2007) mapped studies concerned with combining software product lines with agile software development practices. They emphasized the explicit adherence to the principles of the manifesto for agile software development [2] and identify Scrum and XP (eXtreme Programming) to be the most commonly mentioned methodologies in relation to software product lines.

To our best knowledge, a systematic review of the general circumstances to which Scrum-based development has been tailored is not available. This work therefore aims to close this gap and presents an overview of emerging themes identified in relevant literature. Previous reviews can be situated into the classification presented in this review.

4. Research Method

We followed the widely accepted literature review guidelines outlined in [48]. As our research focus was to examine the literature on adaptations or modifications of Scrum, we defined several keywords to capture relevant studies. In order to increase our understanding of the subject matter and devise a meaningful search strategy [4], we first read and discussed a number of highly cited articles, in combination with insights from related literature reviews (as discussed earlier). We made sure to allow for inclusion of both problem-driven as well as solution-driven initiatives.

To discover relevant literature, we used the Scopus database and followed an iterative process to construct a replicable research query combining the words “Scrum”, “agile”, and “software” as mandatory elements, combined with a range of optional terms targeted to find adaptations of Scrum

both in negative and positive terms. An overview of our final search terms is given in Table 1.

The first search returned a relatively large number of studies (1046). We excluded pure discussion/opinion papers and literature reviews from our analysis, but kept them for discussion and additional insights. Moreover, we applied category-based filter on Scopus, so that papers dealing with sport, rugby, and medicine were not included.

Table 1: Keyword specification for literature search (* = wildcard)

Main topic specification
Scrum, Software, Agile
Negative terms
limit*, drawback, shortcoming, challeng*, concern*, downside*
Neutral terms
demand*, requirement*, need*, issue*, suit*, accommodate*, modif*, tailor*, alter*, adapt*, chang*
Positive terms
exten*, enhanc*, expand*, widen*, improve*, focus*, revis*, fit*, scop*

To filter the search results, our main criteria were quality and practical relevance [34]. An initial screening of the literature indicated the need for a quality cut-off, as many studies were of low scientific quality and described trivial system implementations with no relevant insights.

Table 2: Literature filter process

Raw results of the query: 1046
Filter: <ul style="list-style-type: none"> All articles before 2016 with 10 or more citations (83) All articles in 2016 (15) All articles in 2017 (7) <p>Remaining sample: 105 papers</p>
After screening of title and abstracts: 61
Remaining after full reading: 31 (final sample)

Thus, we devised a three-step filter process (see Table 2), depending on the time of publication and the citation count at the time of our research. First, articles published before 2016 with ten or more citations were included. Second, papers published in 2016 with at least one citation were included. Third,

studies published in 2017 or in print were included regardless of citation count to allow newly published articles to be assessed. This resulted in a preliminary sample of 105 papers, i.e. ~10 per cent of the initial search results. To reduce the chance that relevant papers were excluded by accident, we performed a screening of 100 random articles out of 941 excluded articles. None of the screened studies was included, based on quality and relevance criteria.

Next, the remaining 105 studies were screened based on titles and abstracts by both researchers individually. Differences in coding were resolved by discussion. When in doubt, the paper in question was kept until the next, more thorough, round.

5. Descriptive results

In line with previous reviews, our sample shows that the dominant part of the literature consists of empirical papers. This includes industry reports by practitioners as well as research reports by academics who describe the development practices of selected case organizations. Case study designs are by far the most commonly employed research strategy. Rarely did studies in our selection provide theoretical backing for the proposed adjustments.

We included both journal articles and conference papers in our review. The relatively high amount of conference papers in our sample points to the practical orientation as well as the emerging nature of the topic. A negative consequence of this practical orientation of the available literature is lowered generalizability of the findings.

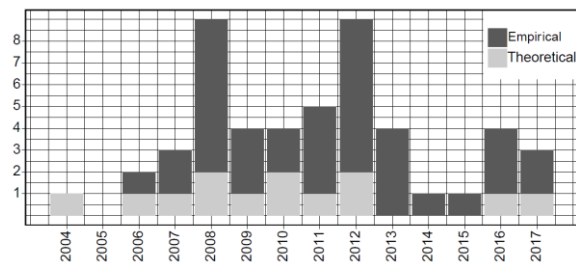


Figure 1: Article types over time

While the first relevant papers we found were published in 2006, the peak interest in this topic can be observed in 2008 and later in 2012 (Figure 1).

The case studies were set in a variety of different industries, with some emphasis on IT companies. While most cases discussed smaller IT companies, some large corporations were also represented, such as PayPal [5], Ericsson [19], and Intel [12]. Other studies focused on rather specialized areas or industries, such as the cruise line industry [1] or

Table 3: Overview of motivations and solutions

right: solutions down: motivations	combination	pre- development	method guidance	procedures, artifacts, roles	multiplicity	tools
distributed	0	0	6	2	5	0
combination	7	4	2	0	0	0
UX and usability	0	1	2	3	3	0
vertical scaling	0	0	0	3	3	0
size scaling	0	1	1	2	3	0
tools	0	0	0	1	0	2
context	0	0	1	2	0	0

healthcare [16]. In terms of geographical dispersions, most studies focus on Western Europe, the Nordic regions, and the United States.

6. Motivation and proposed adaptations

In lack of an existing organizing framework that could support our analysis, we took inspiration from the constant comparative method used in Grounded theory [6], and engaged in a process of coding the articles to generate an inductive frame from within the data. To guide our analysis, we aimed for the discovery of categories along the two main questions that motivate our research: why was the method modified (dimension: motivations) and how was the method modified (dimension: solutions). Through multiple iterative coding sessions in which both authors participated, we iteratively developed categories along these two dimensions. The coding sessions were categorized by alternating discovery and discussion parts, ultimately leading to our categorization frame, as described below.

First, we identified 7 distinct types of motivation for modifying Scrum from its original version: **distributed** settings, **combination** with other frameworks or methods, increased focus on **UX and usability**, **vertical scaling** (i.e., embedding Scrum in larger organizational aspects, such as strategic planning), **size scaling** (i.e., Scrum for medium and large projects), **tools** to use with Scrum, and Scrum in a specific **context**. While some studies relate to more than a single motivation for change, most of the examined articles correspond to a single main motivation in our categorization scheme. Second, similar to the motivations for change, we were able to

identify 6 different solution strategies applied to achieve the intended goal (as described in the next Section).

Both categorization schemes are summarized in a comprehensive matrix (see Table 3), which links the motivations for change with the proposed solution strategies.

To guide the reader through our findings, the following section opens with a brief overview and discussion of the types of modifications we found. Thereafter, we iterate through the different motivation categories in detail, to present and discuss the relevant papers and their solution strategies.

6.1. Solution strategies

The following 6 solution strategies were found in the literature:

- **Combination:** An intermixing of Scrum elements with elements of other existing processes/methods such as CMMI (Capability Maturity Model Integration), Lean, or XP.
- **Pre-development:** This category is a special case of the aforementioned “Procedures, artifacts, roles” category. It refers to the introduction of additional processes, artifacts, or roles that specifically deal with tasks such as the definition of technical architecture, articulating a product vision, or creating milestones for development, before the development itself is initiated.
- **Method guidance:** Notes, instructions, and guidance on how to apply the method in specific settings, contexts, or circumstances. This category includes appeals to “by the book applications” of the selected method, reminders of the principles of the Manifesto for Agile Software Development [2], and

even practical advice in the form of best practice guidelines.

- **Procedures, artifacts, roles:** Change of existing or introduction of new artifacts, roles, or processes to the original Scrum method.
- **Multiplicity:** Multiplication of certain aspects of Scrum (artifacts, processes, roles, or the team itself). The multiplied elements of Scrum can be used for different purposes. For instance, it can be suggested to have two backlogs, one for development and one for management.
- **Tools:** Proposals of tools that do not directly modify Scrum but help accomplish certain task. Such tools can often be seen as a kind of “plug-ins” to the original method, and they may be applied passively without directly changing the method in its workings.

6.2. Motivations for change

6.2.1. Distributed Setting

Distributed settings	
method guidance	[3] [21] [27] [38] [36] [37]
procedures, artifacts, roles	[21] [27]
multiplicity	[21] [27] [38] [36] [37]

In today’s globally connected world, IS development sometimes takes place across different geographical locations, in so-called distributed teams; this can range from teams scattered across continents to teams which are in the same country (or even city). In such setups, communication usually relies on technology-mediation, i.e., the use of video conferencing tools or similar technologies.

Using Scrum in distributed settings usually requires some degree of multiplicity. The Scrum team is often split into several Scrum teams in different locations. In the reviewed literature, the newly formed teams were always split according to specific features (feature-driven), which is in line with the original design of Scrum, rather than being built around a single capability (such as front-end or back-end development). While Scrum teams are usually multiplied in the different locations, the supporting architecture does not need to be redundant. For example, Lee and Yong (2010) report on a team which maintained a global platform with shared backlogs accessed by multiple local teams [27]. A similar practice of a shared backlog is reported in [3][3].

For successful application of Scrum in distributed settings, research emphasizes the need for proper implementation of the method with close adherence

to the principles of the manifesto for agile software development [2]. This is well captured by Paasivaara, Lassenius, and Heikkilä (2012), who quote a manager saying “I think that the first thing is that if you decide to do it, then you need to do it properly. You cannot start using Scrum or agile half-way, [because] then you won’t be able to take out the benefits” [39].

The importance of understanding and adhering to the basic agile practices is a reoccurring theme in the literature on distributed settings. For example, Berczuk (2007) urges practitioners to “ensure that all team members understand and embrace the values of your agile method” [3]. The same paper is also in favor of co-locating the developers together at least for the first sprint, to ensure the development of some form of a trust relationship among the teams.

A number of previous studies focus exclusively on global software engineering [23]. Further, the conceptual framework proposed by [21] offers a number of strategies and practices to mitigate 7 common risks of distributed agile software development, such as using online Wiki’s for key document sharing to mitigate the lack of group awareness, or ensuring a suitable set of communication tools for the available network infrastructure.

6.2.2. Combination

Combination	
combination	[9] [12] [18] [29] [31] [43] [50]
pre-development	[9] [18] [22] [50]
method guidance	[22] [47]

Combining Scrum with other methodologies is a topic receiving a significant attention in the literature.

Some studies argue for an underlying goal behind their combination efforts (i.e., increased efficiency), others simply aim to assess the possibility of their co-existence while identifying potential synergies that can be gained through meaningful combination [31].

Most notable sources of inspiration were the CMMI framework, XP, and Lean development. Elements brought into Scrum frequently provide pre-development activities—such as specification of high-level technical infrastructure—and generally equipped Scrum with more rigidity.

For example, Diaz, Garbajosa, and Calvo-Manzano (2009) find that CMMI level 2 aligns well with Scrum, and that this combination produces positive synergies even for small businesses [9]. A similar conclusion is reached for mature organizations with CMMI level 5 certification [22]. A more comprehensive mapping study between Scrum and the CMMI model is provided in [29].

Studies examining Scrum with CMMI were also mapped by a separate literature review [35].

Generally, CMMI is found to be beneficial for requirements elicitation, budgeting, and risk management, in addition to providing a signaling value of the certification. If implemented right, it allows to “balance agility and discipline” [29] of both methods.

In their study, Van Waardenburg and Van Vliet (2013) report on possible mitigation strategies to deal with the challenges of co-existing plan-driven and agile methods in organizations [50]. They identify the two main factors as “Increased IT Landscape complexity” and “Lack of Business involvement” as a result to the co-existence, and discuss several strategies (contingents) to address these aspects.

Harvie and Agah (2016) include pre-development processes in their flavor of Scrum by drawing inspiration from military theory [18]. They develop a mechanism to support a more formal approach towards managing backlogs, which relies on a so-called “product end state document” that serves as a prioritization guide.

As an overview, Wang, Conboy, and Cawley (2012) provide a review of thirty experience reports about attempts to combine agile and Lean software development, identifying six unique types: non-purposeful combinations; agile within, Lean outreach; Lean facilitating agile adoption; Lean within agile; from agile to Lean; and synchronization of agile and Lean [47].

6.2.3. UX and Usability

UX and Usability	
pre-development	[44]
method guidance	[11] [25]
procedures, artifacts, roles	[11] [25] [44]
multiplicity	[5] [11] [41]

Studies discussing the incorporation of user experience design often suggest establishing two Scrum teams: one for developers and one for designers. Budwig, Jeong, and Kelkar (2009) recommend to “organize the UX team into a separate Scrum team, with its own product backlog and product owner” [5]. They further suggest that the Scrum team proceeds with the work for one or two Scrum iterations ahead of development. For this purpose, the Scrum roles need to be adjusted to accommodate the design tasks, resulting in new roles such as Usability Product Owner in the so-called “U-Scrum” methodology [41].

A risk of separating the designers and programmers is reduced contact between the two

teams and the users. It is important for “team members responsible for Usability and UX [to] have face-to-face communication with the actual users at least once during each sprint.” [25]. Ferreira, Sharp, and Robinson (2011) report on challenges of the communication process between development and design teams, highlighting the differences between the different work sub-cultures [11]. Finally, an experience report by Ungar and White (2008) presents the design practice of a design workshop, in which the stakeholders (developers, managers, customer) are brought together to work on low-fidelity prototypes to clearly establish a shared vision before the development itself is commenced [44]. This is an example for a possible pre-development activity.

The proposed methodology adjustments come with many practical implementation tips. Such method guidance tidbits include recommendations such as “Define measurable goals for Usability” and “Define the responsibility for Usability and UX for all roles” [25].

6.2.4. Vertical Scaling

Vertical Scaling	
procedures, artifacts, roles	[19] [30] [46]
multiplicity	[19] [30] [46]

Scrum, in its original form, offers tools for management of requirements only on the lowest level. Higher levels of software product management such as road mapping [49] and establishing a connection to a firm’s overall strategy are not covered by Scrum. For small teams, it is possibly to duplicate the Scrum process for product management, with the product manager maintaining their own backlog [46]. In larger development efforts, the multiplicity of elements can be nested in Scrum-of-Scrums like architectures [30]. A comprehensive methodology adjustment has been proposed by Vlaanderen et al. (2011), demonstrating a process for translating strategic requirements into features, epics, and stories of agile development process in a large organization with a multitude of Scrum-inspired teams [46]. New artifacts (e.g., a “one-pager” that specifies a feature) are introduced. The methodology also describes new roles (e.g., Chief Product Owner) and processes (e.g., process development).

6.2.5. Size Scaling

Size Scaling	
pre-development	[32]
method guidance	[39]
procedures,	[19] [32]

artifacts, roles	
multiplicity	[39] [19] [32]

Agile development is best suited to small teams, but the complexity of some software products mandates a large number of developers. Papers in this stream offer solutions to managing agile development when the number of developers exceeds what is recommended for a single agile team.

When a multitude of teams is established, they are then often arranged in a “nested” setting, sometimes referred to as Scrum-of Scrums. Infrastructure for team communication across teams usually mirrors the basic Scrum, except that instead of individuals, team representatives are participating on the meetings. When such teams have too many participants, they risk a lack of common interest and knowledge across teams [39]. A recommended practice is therefore to hold Scrum-of-Scrum meetings with fewer participants with joint interests [39].

An alternative framework for organization of large-scale development is the CAFFEA (Continuous Architecting Framework For Embedded software and Agile) framework [32]. In CAFFEA, dedicated roles are created for architecture development and governance. Teams are cross-functional and arranged alongside specific features. The framework puts emphasis on achieving architectural consistency in large-scale software development efforts employing agile methodologies.

6.2.6. Tools

Tools	
procedures, artifacts, roles	[7]
tools	[28] [42]

Several papers are motivated by the need for techniques that do not directly modify the Scrum methodology, but can be used in conjunction with existing Scrum elements to achieve a specific task. Papers in this category are motivated by a need to develop a tool and deliver that tool as a solution. “Tools” is therefore listed as both as a motivation for change as well as a solution.

For example, to improve requirements scheduling, Li et al. (2010) develop a linear programming model and showcase a prototypical application for release planning. The authors show that their scheduling model can be applied for Scrum projects, and may increase planning efficiency among multiple sprints and teams.

From a financial planning perspective, Sulaiman, Barton, and Blackburn (2006) develop AgileEVM – a set of formulae to calculate Earned Value

Management (EVM) parameters for agile projects [42].

In Codabux and Williams (2013), a taxonomy of technical debt is developed based on qualitative research [7]. The authors suggest refactoring, repackaging, and reengineering as activities to reduce technical debt. Suggested practices include the establishment of teams who focus solely on reducing technical debt, as well as dedicating 20% of development time towards the reduction of technical debt.

6.2.7. Context

Context	
method guidance	[24]
procedures, artifacts, roles	[16] [14]

Recently, some authors began to describe cases of the introduction of Scrum to non-traditional contexts. For example, Könnölä et al. (2016) report on successful adoption of Scrum to embedded system development [24]. They provide method guidance highlighting the specific needs of this context, such as longer iteration cycle of hardware development compared to software development. They find that agile development for embedded systems yields numerous benefits, such as clearer dependencies of individual modules among each other.

Fitzgerald et al. (2013) present a heavily modified version of Scrum for environments characterized by heavy regulation, introducing an exhaustive set of new processes, artifacts, and roles.

Finally, Gary et al. (2011) offers a case study on a specific development effort of an open source tool for the healthcare industry. Similarly, this case study also recommends modifying Scrum by adding new processes, roles, and artifacts.

7. Discussion, implications and future research

This study maps a variety of ways in which Scrum has been modified to better fit commonly encountered circumstances. The modifications are categorized into several generic solution strategies, each of which carries certain risks and challenges.

7.1. Scrum in practice

This literature review confirms that Scrum’s software development principles are widely applicable and beneficial in various, often non-traditional settings.

As the literature suggests however, the development methodology and techniques have to be tailored to specific needs of the given circumstances [33]. In many cases, this requires a modification of existing and/or introduction of new roles, processes, and artifacts. However, organizations need to carefully orchestrate new elements to fit with the existing method, as they risk diluting the benefits that an adoption of agile principles promised in the first place. For example, the suggestion of a “product end state document” in [18] may be at odds with the principle of welcoming changing requirements, a core element of the agile manifesto [2].

Another commonly used approach to adapt to changing work practices is the multiplication of the whole method in the form of multiple Scrum teams. Extant literature suggests that this strategy can be useful for many purposes apart from geographically distributed development settings, such as large but co-located project teams, or feature driven Scrum teams that focus on UX and usability topics in parallel to a development team. To reap the benefits of a multiplied Scrum setup, research emphasizes the need to establish well working interfaces between the teams [25], as well as to develop a common work culture [11].

While tailoring of Scrum may take one of many potential forms, extant research stresses the importance of the basic principles that guide agile software development [2]. Interestingly, our review shows that both knowledge of and adherence to those principles become even more important with increasing distance from the originally intended setting of small, collocated, self-managed software development teams [3, 39]. Those principles are likely to be better internalized by highly mature teams who have worked in agile manner for some time. Consequently, adopting a modified Scrum development approach by a newly formed or distributed team may be a risky endeavor.

Interestingly, the combination of Scrum with other frameworks or methods is usually not driven by a limitation of Scrum, but rather a “desire to explore” the potential of infusing some level of agility into—often large and rigid—traditional organizations [18, 31]. Thus, these studies often do not represent a modification of Scrum, but rather an extension of other frameworks (i.e., CMMI) with elements from Scrum. Conversely, the current body of knowledge largely lacks insights into how some of the commonly mentioned challenges for the application of agile methods (i.e., large-scale projects or distributed development) may be solved through systematic “borrowing” from, or combination with, other frameworks.

7.2. Methodological considerations

Our descriptive results show that the majority of the available literature is driven by practitioner interest and activities, thus often taking the form of case studies. Consequently, little research provides sound theoretical backing or links the researched practices to extant theory. Further, due to the predominant single case study design, many reported findings lack statistical generalizability, but provide grounds for analytical generalization [26]. To allow for comparative analyses and increased external validity, we recommend future research to employ multiple case study designs [51]. Moreover, many studies do not follow the academic practice of iterative, cumulative knowledge development, i.e., insufficiently relate their research to the existing knowledge base. Thus, our study may also serve as a frame for more structured future research, encouraging a cumulative research tradition.

8. Conclusion and limitations

The use of agile methods has become a widespread practice for software development teams. As one of the most widely implemented agile methods, Scrum has been the focus of a number of adaptations and modifications. In this study, we provide an in-depth review and synthesis of academic literature proposing changes to Scrum. By analyzing 31 relevant studies, we extract seven distinct motivations for method modifications: distributed settings, combination with other methods, increased requirements for UX and usability, vertical scaling, size scaling, tools, and adaption to different contexts. Additionally, we could identify six generic strategies of how these goals can be achieved: through combination, pre-development, method guidance, introduction of new procedures/artifacts/roles, multiplicity of some method elements, or by developing specific tools. Combined, we present a model of common drivers for method improvement and the respective solutions strategies pursued.

We conclude with some limitations of this study. While we conducted a systematic literature search based on key words, we most likely missed a proportion of relevant literature in particular in terms of publications not listed in the Scopus database. In addition, the use of citations as a quality threshold should be considered with caution. Citations may also signal political biases, alliances and omissions, and be biased towards seminal studies representing “concept labels” [17]. They can also be interpreted as a reflection of the different power relations that

surround a field [15]. Finally, we acknowledge that our review is limited to academic contributions, and thus turns a blind eye towards potentially relevant publications in various non-indexed practitioner outlets, such as blog-posts, discussion forums, and the like.

9. References

- [1] Batra, D., Xia, W., van der Meer, D., and Dutta, K. Balancing agile and structured development approaches to successfully manage large distributed software projects: A case study from the cruise line industry. *Communications of the Association for Information Systems* 27, 1 (2010), 379–394.
- [2] Beck, K., Beedle, M., Bennekum, A. Van, et al. Manifesto for Agile Software Development. *The Agile Alliance* 2009, 2001, 2006. <http://agilemanifesto.org/>.
- [3] Berczuk, S. Back to basics: The role of agile principles in success with an distributed scrum team. *Proceedings - AGILE 2007*, (2007), 382–387.
- [4] vom Brocke, J., Simons, A., Riemer, K., Niehaves, B., Plattfaut, R., and Cleven, A. Standing on the shoulders of giants: Challenges and recommendations of literature search in information systems research. *Communications of the Association for Information Systems* 37, 1 (2015), 205–224.
- [5] Budwig, M., Jeong, S., and Kelkar, K. When user experience met agile. *Proceedings of the 27th international conference extended abstracts on Human factors in computing systems - CHI EA '09*, (2009), 3075.
- [6] Charmaz, K. *Constructing grounded theory: a practical guide through qualitative analysis*. 2006.
- [7] Codabux, Z. and Williams, B. Managing technical debt: An industrial case study. *2013 4th International Workshop on Managing Technical Debt (MTD)*, (2013), 8–15.
- [8] Conboy, K. Agility from First Principles: Reconstructing the Concept of Agility in Information Systems Development. *Information Systems Research* 20, 3 (2009), 329–354.
- [9] Diaz, J., Garbajosa, J., and Calvo-Manzano, J.A. Mapping CMMI level 2 to scrum practices: An experience report. *Communications in Computer and Information Science* 42, (2009), 93–104.
- [10] Duechting, M., Zimmermann, D., and Nebe, K. Incorporating user centered requirement engineering into agile software development. *Proceedings of the HCII 2007*, (2007), 58–67.
- [11] Ferreira, J., Sharp, H., and Robinson, H. User experience design and agile development: Managing cooperation through articulation work. *Software - Practice and Experience* 41, 9 (2011), 963–974.
- [12] Fitzgerald, B., Hartnett, G., and Conboy, K. Customising agile methods to software practices at Intel Shannon. *European Journal of Information Systems* 15, 2 (2006), 200–213.
- [13] Fitzgerald, B., Russo, N., and Stolterman, E. *Information Systems Development: Methods in Action*. 2002.
- [14] Fitzgerald, B., Stol, K.J., O’Sullivan, R., and O’Brien, D. Scaling agile methods to regulated environments: An industry case study. *Proceedings - International Conference on Software Engineering*, (2013), 863–872.
- [15] Foucault, M. What is an Author? *Contributions in Philosophy*, (2001).
- [16] Gary, K., Enquobahrie, A., Ibanez, L., et al. Agile methods for open source safety-critical software. *Software - Practice and Experience* 41, 9 (2011), 945–962.
- [17] Hansen, S., Lyytinen, K., and Markus, M. The Legacy of “Power and Politics” in Disciplinary Discourse: A Citation Analysis. *ICIS 2006 Proceedings*, (2006).
- [18] Harvie, D.P. and Agah, A. Targeted Scrum: Applying Mission Command to Agile Software Development. *IEEE Transactions on Software Engineering* 42, 5 (2016), 476–489.
- [19] Heikkilä, V.T., Paasivaara, M., Lasssenius, C., Damian, D., and Engblom, C. Managing the requirements flow from strategy to release in large-scale agile development: a case study at Ericsson. *Empirical Software Engineering*, (2017), 1–45.
- [20] Hossain, E., Babar, M.A., and Paik, H. Using Scrum in Global Software Development: A Systematic Literature Review. *2009 Fourth IEEE International Conference on Global Software Engineering*, (2009), 175–184.
- [21] Hossain, E., Babar, M.A., Paik, H., and Verner, J. Risk Identification and Mitigation Processes for Using Scrum in Global Software Development: A Conceptual Framework. *2009 16th Asia-Pacific Software Engineering Conference*, (2009), 457–464.
- [22] Jakobsen, C.R. and Johnson, K.A. Mature agile with a twist of CMMI. *Proceedings - Agile 2008 Conference*, (2008), 212–217.
- [23] Jalali, S. and Wohlin, C. Agile practices in global software engineering - A systematic map. *Proceedings - 5th International Conference on Global Software Engineering, ICGSE 2010*, (2010), 45–54.
- [24] Könnölä, K., Suomi, S., Mäkilä, T., Jokela, T., Rantala, V., and Lehtonen, T. Agile methods in embedded system development: Multiple-case study of three industrial cases. *Journal of Systems and Software* 118, (2016), 134–150.
- [25] Larusdottir, M., Gulliksen, J., and Cajander, Å. A license to kill – Improving UCSD in Agile development. *Journal of Systems and Software* 123, (2017), 214–222.
- [26] Lee, A.S. and Baskerville, R.L. Generalizing Generalizability in Information Systems Research. *Information Systems Research* 14, 3 (2003).
- [27] Lee, S. and Yong, H.-S. Distributed agile: project management in a global environment. *Empirical Software Engineering* 15, 2 (2010), 204–217.
- [28] Li, C., van den Akker, M., Brinkkemper, S., and Diepen, G. An integrated approach for requirement selection and scheduling in software release planning. *Requirements Engineering* 15, 4 (2010), 375–396.
- [29] Łukasiewicz, K. and Miler, J. Improving agility and discipline of software development with the Scrum and CMMI. *IET Software* 6, 5 (2012), 416.
- [30] Laanti, M. Implementing program model with agile principles in a large software development organization. *Proceedings - International Computer Software and Applications Conference*, (2008), 1383–1391.

- [31] Marcal, A.S.C., Freitas, B.C.C., Soares, F.S.F., Furtado, M.E.S., Maciel, T.M., and Belchior, A.D. Blending Scrum practices and CMMI project management process areas. *Innovations in Systems and Software Engineering* 4, 1 (2008), 17–29.
- [32] Martini, A. and Bosch, J. A multiple case study of continuous architecting in large agile companies: current gaps and the CAFFEA framework. *Proceedings - 2016 13th Working IEEE/IFIP Conference on Software Architecture, WICSA 2016*, (2016), 1–10.
- [33] Obwegeser, N. Entrepreneurial IS development: why techniques matter and methods don't. In M. Kurosu, ed., *Human-Computer Interaction: Users and Contexts*. Springer, 2015, 218–225.
- [34] Okoli, C. and Schabram, K. A Guide to Conducting a Systematic Literature Review of Information Systems Research. *Working Papers on Information Systems* 10, 26 (2010), 1–51.
- [35] Palomino, M., Dávila, A., Melendez, K., and Pessoa, M. Agile practices adoption in CMMI organizations: A systematic literature review. *Advances in Intelligent Systems and Computing*, (2017), 57–67.
- [36] Paasivaara, M., Durasiewicz, S., and Lassenius, C. Distributed agile development: Using Scrum in a large project. *Proceedings - 2008 3rd IEEE International Conference Global Software Engineering, ICGSE 2008*, (2008), 87–95.
- [37] Paasivaara, M., Durasiewicz, S., and Lassenius, C. Using scrum in a globally distributed project: A case study. *Software Process Improvement and Practice* 13, 6 (2008), 527–544.
- [38] Paasivaara, M., Durasiewicz, S., and Lassenius, C. Using Scrum in Distributed Agile Development: A Multiple Case Study. *2009 Fourth IEEE International Conference on Global Software Engineering*, (2009), 195–204.
- [39] Paasivaara, M., Lassenius, C., and Heikkilä, V.T. Inter-team coordination in large-scale globally distributed scrum. *Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement - ESEM '12*, (2012), 235.
- [40] Schwaber, K. SCRUM Development Process. In *Business Object Design and Implementation*. 1997, 117–134.
- [41] Singh, M. U-SCRUM: An agile methodology for promoting usability. *Proceedings - Agile 2008 Conference*, (2008), 555–560.
- [42] Sulaiman, T., Barton, B., and Blackburn, T. AgileEVM-earned value management in Scrum Projects. *Proceedings of the Agile Conference, Minneapolis (MN), USA, 23-28 July, 2006*, (2006), 10 pp. pp.16.
- [43] Sutherland, J., Jakobsen, C.R., and Johnson, K. Scrum and CMMI level 5: The magic potion for code warriors. *Proceedings - AGILE 2007*, (2007), 272–277.
- [44] Ungar, J. and White, J. Agile user centered design. *Proceeding of the twenty-sixth annual CHI conference extended abstracts on Human factors in computing systems - CHI '08*, (2008), 2167.
- [45] VersionOne. *11th State of Agile Report*. 2017.
- [46] Vlaanderen, K., Jansen, S., Brinkkemper, S., and Jaspers, E. The agile requirements refinery: Applying SCRUM principles to software product management. *Information and Software Technology* 53, 1 (2011), 58–70.
- [47] Wang, X., Conboy, K., and Cawley, O. “Leagile” software development: An experience report analysis of the application of Lean approaches in agile software development. *Journal of Systems and Software* 85, 6 (2012), 1287–1299.
- [48] Webster, J. and Watson, R.T. Analyzing the Past to Prepare for the Future: Writing a Literature Review ANALYZING THE PAST TO PREPARE FOR THE FUTURE: WRITING A. *MIS Quarterly* 26, 2 (2002), 13–23.
- [49] van de Weerd, I., Brinkkemper, S., Nieuwenhuis, R., Versendaal, J., and Bijlsma, L. Towards a Reference Framework for Software Product Management. *RE'06: 14th IEEE International Requirements Engineering Conference*, (2006), 319–322.
- [50] Van Waardenburg, G. and Van Vliet, H. When agile meets the enterprise. *Information and Software Technology* 55, 12 (2013), 2154–2171.
- [51] Yin, R.K. *Case Study Research: Design and Methods*. Sage Publications, 2009.