Do the Scaled Agile Practices from S@S Help with Quality Requirements Challenges and If So, How Do They Do It?

Wasim Alsaqaf, Maya Daneva and Roel Wieringa
School of Computer Science, University of Twente, Enschede, The Netherlands
{w.h.a.alsaqaf, m.daneva, r.j.wieringa}@ utwente.nl

Keywords: Agile Scaled Framework, Scrum@Scale, S@S, Quality Requirements, Requirements Engineering, Non-functional Requirements, Documentary Research Method.

Abstract: Quality Requirements (QRs) pose challenges in many agile large-scale distributed enterprise systems. Often, enterprises counter such challenges by borrowing some heavyweight practices, e.g. adding more documentation. At the same time, agile methodologists proposed several scaled agile frameworks to specifically serve agile enterprises working on large and distributed systems. Little is known about the extent to which the proposed scaled frameworks address QRs and the specific ways in which this happens. Moreover, do these frameworks approach the QRs challenges in ways consistent with the Agile Manifesto? This paper treats these questions by analyzing one well-documented scaled framework, namely Scrum@Scale. We evaluated the alignment of Scrum@Scale with the Agile Manifesto, by means of the 4-Dimentional Analytical Tool proposed by other researchers. We then analyzed the practices of Scrum@Scale from the perspective of practitioners responsible for the QRs in a project, in order to understand how the Scrum@Scale practices mitigate those QRs challenges reported in previous work. Our analysis indicated that Scrum@Scale supports the agile values defined by the Agile Manifesto. Plus, we identified 12 Scrum@Scale practices that could (partially) mitigate one or more of the reported QRs challenges. Four of the reported QRs challenges got no remedy offered by Scrum@Scale.

1 INTRODUCTION

Currently, the software market is marked by two strong trends: agile and distributed (Calefato and Ebert, 2019). Both are increasingly more demanded in large-scale project delivery (Smart, 2018). However, the transferability of experiences made in the original context for which agile development methods were originally designed – small, co-located teams – to the realities of large-scale distributed contexts is far from flawless (Smart, 2018; Conboy and Carroll, 2019; Kalenda, Hyna, and Rossi, 2018; Bick et al., 2018). Agile methodologists do provide guidelines to enterprises on how to transform to large-scale distributed agile, which often come in the format of the so-called 'agile scaled frameworks', e.g. Disciplined Agile Delivery (DAD) (Ambler and Lines, 2012) among others. However, as Smart (2018) observes, relatively little research is published about these frameworks' effectiveness in practice, especially on an enterprise scale. Moreover, as per a 2018 review (Kalenda, Hyna, and Rossi, 2018), large-scale agile enterprises adopting these frameworks report a broad range of technical and enterprise-level challenges due to resistance to change, shifts in the ways of thinking of hierarchies of requirements, lack of transparency, and lack of knowledge on proper integration of agile and non-agile ways of working. This paper is dedicated to one specific type of requirements challenges in large-scale agile delivery, namely those pertaining to QRs, such as security and usability. The paper builds upon an earlier study (Alsaqaf, Daneva, and Wieringa, 2019) in which the authors found that often, enterprises counter QRs challenges by borrowing some heavyweight practices, e.g. creating new artefacts (security or usability stories) or roles (e.g. security officer, User Experience team), and then adding these practices to their agile delivery cycle. Therein (Alsaqaf, Daneva, and Wieringa, 2019), is also stated that the introduction of these heavyweight practices unexpectedly brought with them new problems. But do agile methodologists propose to remedy QRs issues in large-scale agile, by injecting more heavyweight practices in the development process, and, eventually, making it less agile? Do these

Alsaqaf, W., Daneva, M. and Wieringa, R.
Do the Scaled Agile Practices from S@S Help with Quality Requirements Challenges and If So, How Do They Do It?.
DOI: 10.5220/0010514304410452
In *Proceedings of the 16th International Conference on Software Technologies (ICSOFT 2021)*, pages 441-452
ISBN: 978-989-758-523-4

proposed frameworks approach the QRs challenges in ways consistent with the Agile Manifesto (Agile Alliance, 2001)? As we found no publication answering these questions, we initiated a documentary research process to understand and evaluate the methodologists' proposals for treating QRs challenges. For the purpose of our research we chose for inclusion those scaled agile frameworks deemed 'most popular' according the 14th annual state of agile report (COLLAB.NET and VERSIONONE.COM, 2020).

As already said, the present work rests on a previous published exploratory study (Alsaqaf, Daneva, and Wieringa, 2019) that found 15 QRs challenges and 9 practices that agile practitioners currently use to cope with the identified challenges. We note that these findings (Alsaqaf, Daneva, and Wieringa, 2019) came out of an interview-based research with practitioners in enterprises committed to agile project delivery. However, these 9 practices were not collected in relation to any existing prescriptive or descriptive agile scale framework such as DAD (Ambler and Lines 2012) nor agile method such as Scrum (Schwaber and Sutherland, 2017). Given this background, in the present research we aim to explore those agile practices that are suggested by the most popular published agile scaled frameworks and that could help mitigate the QRs challenges which were identified in our previous work (Alsaqaf, Daneva, and Wieringa, 2019). Particularly, we want to know those practices designed by agile-at-scale methodologists that are agile in nature and align with the values of the Agile Manifesto and not heavyweight practices that when added to an agile process have a tendency to make it less agile.

The present paper reports our results of analysing one specific scaled framework, namely, Scrum at Scale (S@S) proposed by Sutherland (2019). Our selection of S@S is explained later in section 2.1. Here, we would like to note that our ongoing research includes also some other frameworks, however these are out of scope in this paper. This being said, in the research that we report in the present paper, we set out to answer the following research question: *What are the agile practices suggested by the S@S agile scaled framework that could mitigate the effect of the QRs challenges identified in (Alsaqaf, Daneva, and Wieringa 2019) ?* Using a documentary research process (Appleton and Cowley, 1997; Bowen, 2009; Atkinson and Coffrey, 2004), we analyzed the practices that the S@S methodologist (Sutherland, 2019) proposed to use in large enterprises projects. We first evaluated the alignment of S@S with the Agile Manifesto, by means of the 4-Dimentional Analytical Tool (4-DAT) proposed by other researchers (Qumer and Henderson-Sellers, 2006; Qumer and Henderson-Sellers, 2008). We then analysed the practices of S@S from the perspective of practitioners responsible for engineering the QRs in a project, in order to understand how the S@S practices mitigate those QRs challenges reported in previous work. In what follows, we first describe our research process and provide definitions of the most important concepts (Sect. 2 and Sect. 3). We then present our results (Sect. 4) and our discussion on our findings (Sect. 5), on the limitations of this research (Sect. 6) and on its implications (Sect. 7).

2 RESEACH PROCESS

The overall aim of our research is to investigate the agile practices suggested by published agile scaled frameworks which could mitigate the impact of the QRs challenges which were identified in a previous study (Alsaqaf, Daneva, and Wieringa 2019). Towards this end we set up a research process inspired by the documentary analysis methodologists Appelton and Cowley (1997), Bowen (2009) and Attkinson and Coffey (2004). We chose these methodological guidelines because of their suitability to our research context. As Appleton and Cowley state, documentary research is defined as the research conducted through the use of official documents as the source of information. And it is the official documents of scaled agile (i.e. guidelines in the textbooks on scaled agile) that we want to examine in our research context. As Figure 1 shows, our research process included these steps: (1) selecting agile scaled frameworks for inclusion in the research, (2) selecting an agile analysing tool to asses the degree of agility, (3) evaluating their degree of agility, and (4) evaluating the extent to which the practices proposed in the frameworks mitigate the QRs challenges (Alsaqaf, Daneva, and Wieringa 2019).

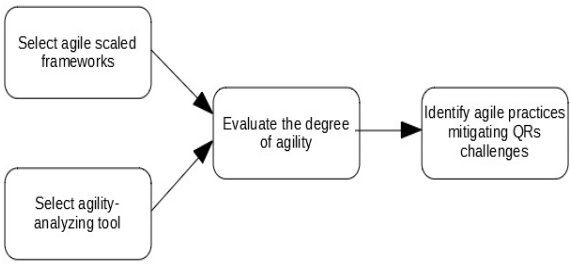

Figure 1: Our research process.

Step (1) explains our reasoning for including certain frameworks. Step (2) explains our reasoning for choosing a certain agility analysing tools. Step (3) is concerned with the evaluation of how agile a scaled framework is, as proposed by its authors in their framework's documentation (and not as implemented in a particular organization). Step (4) is concerned with the matching of the agile practices proposed by the S@S authors against the QRs challenges found in a previously published study (Alsaqaf, Daneva, and Wieringa, 2019). As this paper is focused on one framework only (S@S (Sutherland, 2019)), it in turn reports on steps 2 and 3 as executed in the context of analysing this specific framework. We describe the steps of our process in the next sub-sections.

2.1 Selecting Agile Scaled Frameworks

Portman (2017) has reported the existence of more than 30 agile scaled frameworks and classified these into two categories, namely (1) Enterprise-targeted frameworks (e.g. SAFe (Leffingwell and Knaster 2017), LeSS (Larman and Vodde, 2016), Nexus (Schwaber 2018), S@S (Sutherland 2019)) which are used to deliver complex enterprise-level products whereby the collaboration between distributed teams is essential and (2) Web scale-targeted frameworks (e.g. Spotify (Kniberg and Ivarsson, 2012), Scaled Agile Lean Development (ScALeD)[1]) which are used to support the IT-department of an enterprise in maintaining the existing applications whereby the dependencies between distributed teams are minimalized. In this paper, we focus on the first category of frameworks because these frameworks match our research interest, namely the distributed and large-scale context. Furthermore, we limit our selection of frameworks to those that are the most used according to the 14th annual state of agile report (COLLAB.NET and VERSIONONE.COM, 2020). These sources indicated the following agile scaled frameworks as the most popular: 1) SAFe (Leffingwell and Knaster, 2017), 2) SoS (Sutherland 2001), 3) Internally created methods, 4) DAD (Ambler and Lines, 2012), 5) LeSS (Larman and Vodde, 2016), 6) Enterprise Scrum (ES) (Beedle 2018), 7) Lean management[2], 8) Agile Portfolio Management (AgilePM) (Krebs, 2008), 9) Nexus (Schwaber 2018), 10) Recipes for Agile Governance in the Enterprise (RAGE)[3]. The intersection between the Enterprise-targeted frameworks in (Portman, 2017) and the most popular agile scaled framework described in (COLLAB.NET and VERSIONONE.COM, 2020) reduces our selection group to SAFe (Leffingwell and Knaster, 2017), LeSS (Larman and Vodde, 2016), Nexus (Schwaber 2018), S@S (Sutherland, 2019), SoS (Sutherland, 2001), DAD (Ambler and Lines 2012), ES (Beedle, 2018), AgilePM (Krebs, 2008), Lean management frameworks and RAGE. In this paper, we focus solely on the agile practices of the S@S (Sutherland 2019) framework. We note that S@S is built upon SoS (Sutherland 2001) and Scrum (Schwaber and Sutherland 2017), both of which are among the most used agile frameworks and methods (COLLAB.NET and VERSIONONE.COM, 2020). However, our choice for S@S (Sutherland, 2019) does not mean that we prefer or recommend S@S (Sutherland, 2019). The other frameworks will be investigated in our follow-up research.

2.2 Selecting Agility Analysing Tool

In order to evaluate the degree of agility of S@S (Sutherland 2019) we selected the 4-Dimensional Analytical Tool (4-DAT) described by Qumer et al. (Qumer and Henderson-Sellers, 2006; Qumer and Henderson-Sellers 2008). We note that there are other approaches that assess the agility level of an agile software development framework such as the Conceptual Framework of Agile Methods described by Conboy et al. (Conboy and Fitzgerald, 2004) and the AgilityMod approach of Özcan-Top and Demirors (2019). However, in contrast to the 4-DAT approach (Qumer and Henderson-Sellers, 2006; Qumer and Henderson-Sellers, 2008) which is focused on the agile practices of the agile scaled framework itself, these other assessment frameworks (Conboy and Fitzgerald, 2004; Özcan-Top and Demirors 2019) focus on the agility factor of the particular application of the particular framework's practices within a particular enterprise by agile teams. Moreover, Conboy and Carroll (2019) note that the right implementation of an agile scaled framework by software development teams depends on multiple factors (e.g. a solid understanding of the agile scaled framework, the skills and knowledge of the involved software development teams). In turn, evaluating the agile practices as implemented by software development teams does not give an insight in how the agile scaled framework itself describes its own practices. It merely describes the way the software development teams implement the particular agile

[1] http://scaledprinciples.org/

[2] https://www.lean.org/WhatsLean/

[3] https://www.cprime.com/rage/

scaled framework. Taking into account that the 14th annual state of agile report (COLLAB.NET and VERSIONONE.COM, 2020) has stated (1) Lack of skills/experience with agile methods, (2) Insufficient training and education, and (3) Inconsistent processes and practices across teams, as challenges experienced in scaling agile, we decided to evaluate the practices as described by the authors of S@S (Sutherland, 2019) and the S@S-related literature on the S@S website.

2.3 Evaluating the Degree of Agility

Since the introduction of the Agile Manifesto in 2001 (Agile Alliance, 2001), over 30 frameworks have been published that claim to be agile. Each has based its claim on providing practices that adhere to some or all of the agile principles described in the Agile Manifesto (Agile Alliance, 2001). However, while creating a framework for scaling up agility, it might well be possible that the framework's authors introduce some heavyweight practices into it. This is because scaling up agility necessarily involves some balancing of agility and discipline and of organizational structures and assumed coordination mechanisms and roles (Conboy and Carroll 2019). In fact, a 2018 literature review (Putta, Paasivaara, and Lassenius, 2018) on the adoption of the SAFe framework reports that "moving away from agile" as an important challenge, among others. Evaluating the degree of agility of an agile scaled framework is therefore essential to be able to accept or reject its practices or part of them as agile practices. In our research, we selected 4-DAT in order to evaluate the degree of agility of S@S as mentioned in section 2.2.

2.4 Identifying Practices Mitigating QRs Challenges

The literature on S@S (Sutherland, 2019) in its official website www.scrumatscale.com was investigated. The first two authors analysed the S@S practices based on their description and fitness to mitigate the QRs challenges identified in the previous study (Alsaqaf, Daneva, and Wieringa, 2019). The analysis started with reading and re-reading the reference document of S@S (see ref. (Sutherland, 2019)) and the information on www.scrumatscale.com that pertains to the 12 large enterprises that implemented S@S, which served as input. Both researchers then checked the relevance of each S@S practice for mitigating the QRs challenges in (Alsaqaf, Daneva, and Wieringa, 2019) which are listed in Table 5.

3 BACKGROUND AND DEFINITIONS

3.1 Scrum@Scale (S@S)

S@S is created by a former medical school professor Jeff Sutherland, also known as the co-creator of the original Scrum (Schwaber and Sutherland, 2017). He defines S@S as: *A framework within which networks of Scrum teams operating consistently with the Scrum guide can address complex adaptive problems, while creatively delivering products of the highest possible value*. It is a framework for scaling Scrum. It radically simplifies scaling by using Scrum to scale Scrum. This definition positions Scrum as the fundament that S@S was built upon. Next, Sutherland uses the term 'scale-free architecture' to denote the way in which Scrum evolves toward S@S. He compares that with scaling a single cell (e.g. Scrum) toward a biological organism (e.g. S@S). Therefore S@S emphasizes the creation of a Reference Model at the very beginning of scaling Scrum. The Reference Model is a set of Scrum teams, each of which implements Scrum as defined by the Scrum guide (Schwaber and Sutherland 2017) and evolves toward S@S. S@S includes two cycles, namely: the Scrum Master Cycle accountable for how to implement the system and the Product Owner Cycle which is accountable for what should be implemented.

3.1.1 The Scrum Master Cycle

It describes team-level processes in which Scrum is applied as per the Scrum guide (Schwaber and Sutherland, 2017). At this level all Scrum roles (e.g. scrum master, development team product owner), Scrum events (e.g. Sprint Planning, Daily Scrum, Sprint Re- view, Sprint Retrospective, the Sprint) and Scrum artefacts (e.g. Product backlog, Sprint backlog, and Product increment) are implemented. In contrast to Scrum, S@S emphasizes that the size of a Scrum team must be between 4 and 6 members (as opposed to 3 to 9 members in (Schwaber and Sutherland, 2017)). Moreover, S@S recommends splitting every Scrum team of 6+ people into two teams. The coordination between the Scrum teams at team-level is done by a Scrum of Scrums (SoS) team which is a Scrum team that has all needed skills to coordinate the work among the individual Scrum teams. A SoS team may coordinate the work of up to 5 Scrum teams. Depending on the size of the project's organization, multiple SoS teams may be needed. In that case, a Scrum of Scrum of Scrums (SoSoS) team must be created to coordinate the work of up to 5 SoS

teams as depicted in Figure 2. The Scrum Master of the SoS team is called the Scrum of Scrums Master (SoSM), while the Scrum master of the SoSoS is called Scrum of Scrum of Scrums Master (SoSoSM). Large agile organizations may have multiple SoSoS teams. The work of those teams will be coordinated by the so-called Executive Action Team (EAT) which is the SoS team of the entire agile organization (see Figure 2). The EAT's members must have enough skills and be empowered to enable the right implementation of Scrum within the organization and to remove any impediments of high level that cannot be removed at lower SoS level.

Figure 2: The structure of scrum teams in S@S.

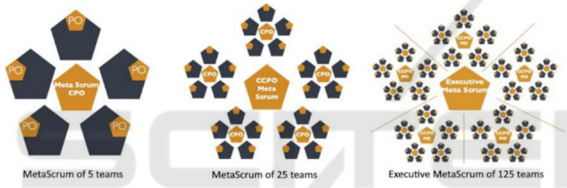

Figure 3: The structure of Product Owner teams in S@S.

3.1.2 The Product Owner Cycle

It assures a clear overview of what is needed to be done during the agile project. Product owner (PO) teams are established in this cycle. The product owners (POs) of each single Scrum team of a particular SoS team are grouped into a PO team that serves the whole SoS that the Scrum teams are part of. In line with the scale-free architecture of S@S, the PO team can grow into a bigger structure in the same way SoS's grow into SoSoS's structure (see Figure 3). However, S@S doesn't provide a name for that bigger structure of PO teams. PO teams at SoS-level as well as SoSoS-level hold frequently a so-called MetaScrum meeting with stakeholders to refine the overall Product Backlog as depicted in Figure 3. Moreover, since the PO team itself is a Scrum team, it has its own Scrum master. Besides the Scrum Master role, the PO team has a new role, namely the Chief Product Owner (CPO). The CPO is responsible for coordinating the work needed to generate the product backlog of the SoS teams that the CPO's PO team is part of. Similarly to the EAT, large agile organizations may set up an empowered PO team for the whole agile organization. Such a team is called in S@S the Executive MetaScrum (EMS) (see Figure 3).

3.2 4-Dimensional Analytical Tool (4-DAT)

This section explains the evaluation model that we use for understanding the degree of agility of S@S. Qumer and Henderson-Sellers (2006 and 2008) have developed the 4-DAT tool to compare agile methods and evaluate their degree of agility in terms of four dimensions.

3.2.1 Dimension 1: Method Scope Characterizations

It serves to compare agile methods at scope level, by checking key scope items (e.g. Project Size, Team Size, Development Style, Code Style, Technology Environment, Physical Environment, Business Culture, Abstraction Mechanism as described) (Qumer and Henderson-Sellers 2008).

3.2.2 Dimension 2: Agility Characterizations

This is a set of agility features to measure the agility of a given method. These features are: flexibility (FY), speed (SD), leanness (LS), learning (LG) and responsiveness (RS). The authors derived these agility features from the following working definition of agility: *"Agility is a persistent behaviour or ability of a sensitive entity that exhibits flexibility to accommodate expected or unexpected changes rapidly, follows the shortest time span, uses economical, simple and quality instruments in a dynamic environment and applies updated prior knowledge and experience to learn from the internal and external environment"*. Dimension 2 is quantitative and is evaluated by identifying the presence or absence of the agility features in high level elements (e.g. phases) and low level elements (e.g. practices) of a given method. The elements are shown in Table 1. Therein, a value of 0 or 1 is assigned to each agility feature (FY, SD, LS, LG and RS, see the respective columns of Table 1), where 0 and 1 mean absence and presence of a feature, respectively. Then, the average of degree of agility can be calculated using the equation provided in (Qumer and Henderson-Sellers 2008).

3.2.3 Dimension 3: Agile Values Characterizations

It evaluates whether the practices of the method to be examined, support six agile values: four of those are

the values provided by the Agile Manifesto (Agile Alliance, 2001), while the other two were reported by (Qumer and Henderson-Sellers, 2006) (see Table 2).

3.2.4 Dimension 3: Software Process Characterizations

This dimension examines those practices of agile methods that support four components of the software development process, namely: (1) Development process, (2) Project management process, (3) Support process, and (4) Process management process.

Table 1: Dimension 2: Agility Characterizations.

Scope item	Description
Flexibility (FY)	Does the method accommodate expected or unexpected changes?
Speed (SD)	Does the method produce results quickly?
Leanness (LS)	Does the method follow the shortest time span, use economical, simple and quality instruments for production?
Learning (LG)	Does the method apply updated prior knowledge and experience to create a learning environment?
Responsiveness (RS)	Does the method exhibit sensitiveness?

Table 2: Dimension 3: Agile Values Characterizations.

Agile values	Description
Individuals and interactions over processes tools	Which practices value people and interaction over processes and tools?
Working software over comprehensive documentation	Which practices value working software over comprehensive documentation?
Customer collaboration over contract negotiation	Which practices value customer collaboration over contract negotiation?
Responding to change over following a plan	Which practices value responding to change over following a plan?
Keeping the process agile	Which practices helps in keeping the process agile?
Keeping the process cost effective	Which practices helps in keeping the process cost effective?

Based on the aforementioned description of 4-DAT, Dimensions 2 (Agility Characterizations) and 3 (Agile Values Characterizations) are applicable for achieving our research objectives stated in Section 2. Dimensions 1 (Method Scope Characterizations) and 4 (Software Process Characterization) are therefore beyond the scope of this paper. Section 4 first describes the practices and phases of S@S that were subjected to our evaluation on Dimensions 2 and 3, and then presents how these practices possibly mitigate those QRs challenges identified in previously published case study (Alsaqaf, Daneva, and Wieringa, 2019).

4 FINDINGS

4.1 Evaluating the Degree of Agility

We investigated S@S as described in its literature and studied its phases and practices. Furthermore, the degree of agility of the S@S's phases and practices was measured in terms of the aforementioned agility features (e.g. FY, SD, LS, LG and RS in Table 1).

4.1.1 S@S Phases

The official literature of S@S (Sutherland, 2019) (e.g. https://www.scrumatscale.com/) doesn't mention particular phases specific to S@S. However, S@S uses Scrum intensively to Scale Scrum teams. From this perspective, we can safely assume that the Scrum phases as described by (Schwaber and Beedle, 2001) are also applied to S@S. The Scrum phases are:

a) Pregame. The Scrum pregame phase consists of two activities, namely, planning and creating high level design. The planning activity concerns with defining project goals, creating the initial product backlog, selecting the product owner, identifying significant architectural and business requirements, and identifying potential risks. After the planning, the identified significant product backlog items get analyzed to create and review the initial system architecture.

b) Game. At this phase the actual development of the system occurs in one to four iterative weeks called 'The Sprint'. The Sprint starts with a planning meeting where product backlog items are selected to be implemented during the Sprint (e.g. Sprint product backlog items). Each Scrum team is responsible for implementing and testing its own Sprint backlog items prior to integrate them as part of the whole system. The output of each Sprint is called 'an incremen't. The cumulative outcomes of all Sprints are a potentially shippable release.

c) Postgame (Closure). At this stage the integrated and tested system (e.g. a potentially shippable release) is stable enough for customer's general release. Final system tests, final user

documentation, user training and marketing activities could be part of this last stage of the Scrum process.

4.1.2 S@S Practices

S@S as described by its literature (Sutherland 2019 and the official website) is a framework where networks of Scrum teams operate consistently with the Scrum guide described in (Schwaber and Sutherland, 2017), to address a complex problems. That means that at the very bottom of the S@S framework, Scrum teams apply Scrum practices (e.g. Sprint, Definition of Done, Sprint retrospective, Scrum master). Those Scrum teams collaborate together by means of additional scaled practices defined by S@S to coordinate the collaboration between the Scrum teams. In this section, we only report and measure the degree of agility of those additional scaled practices as defined by S@S (Sutherland, 2019).

a) Scrum of Scrums (SoS). SoS is a technique to scale Scrum (see Figure 2). It was first described in (Sutherland 2001) by the co-creator of Scrum and creator of S@S Jeff Sutherland. SoS is a Scrum team which is created to coordinate the work of a set of single Scrum teams in order to deliver customer's value. The SoS needs to have all needed skills (e.g. architects, QA experts, Product Owners) to ensure that all parts developed by the different single Scrum teams which are part of the SoS, are fully integrated in a potentially shippable customer's product.

b) Impediment Removal Backlog Artefact. An SoS team maintains its own backlog artefact. Besides product backlog items, the backlog artefact of an SoS contains impediments raised by the Scrum teams that need to be removed.

c) SoS Backlog Refinement Meeting. In this refinement meeting, the representatives of the Scrum teams that make up the SoS team discuss the prioritized impediments on the impediment removal backlog artefact. The impediments that are identified as "ready to be removed" are further explored to determine the most suitable way to remove them and how to confirm their removal.

d) Scaled Daily Scrum. Each SoS team performs its own up to 15 minutes daily Scrum meeting. S@S encourages that representatives of the participating Scrum teams and a representative of the Product Owner team attend this SoS Scaled Daily Scrum. During this meeting the attendees discuss the progress of the Sprint and track the status of the impediments that have been raised by the Scrum teams which may impact the Sprint goal or the upcoming release. Further, the SoS Scaled Daily Scrum is used to improve the collaboration between the participating Scrum teams.

e) SoS Retrospective. Similarly to a Scrum team, an SoS team hold a retrospective meeting. This meeting gives the representatives of the participating Scrum teams the opportunity to share best practices and improve the learning process. Moreover, S@S emphasizes the importance of this meeting as a tool for process improvement.

f) Scrum of Scrums Master (SoSM). The SoSM is part of the SoS team and is responsible for the integration of the completed work of the Scrum teams participating in her SoS. The SoSM is further accountable for enhancing transparency regarding work progress and facilitating the prioritizing of the impediment removal backlog items.

g) Scrum of Scrum of Scrums (SoSoS). When there is more than one SoS team, the work of those teams needs to be coordinated in a structured manner. S@S coordinates the work of multiple SoS teams through a SoSoS team (see Figure 2). A SoSoS team interact with the SoS teams participating in it in the same way in which a SoS team interact with Scrum teams participating in that particular SoS team. Further, a SoSoS team itself is a Scrum team and need to apply the Scrum guide (Schwaber and Sutherland 2017) like any other Scrum team. The number of SoSoS teams can grow infinitely depending on the number of Scrum teams an organization has.

h) Executive Action Team (EAT). In S@S, the EAT is the SoS of the entire enterprise. It coordinates the work of multiple SoS's or multiple SoSoS's. The EAT team is a Scrum team as well and consists of empowered people who can makes financial and strategic decisions. The EAT is responsible for transforming the enterprise into a fully agile one. Further, the EAT is the last resort for escalating and resolving those impediments that cannot removed by lower level SoS's.

i) Product Owner Team (PO team). In S@S each Scrum team has a PO. The group of product owners of the Scrum teams belong to one SoS forms together a PO team of that particular SoS. The PO team is a Scrum team as well and need to adhere to the Scrum guide. Further, the PO team is responsible among others for prioritizing the product backlog of the associated SoS, defining a shared Definition of Done, making technical debts visible in the product backlog and planning the upcoming release. It is also responsible for coordinating the work that needs to be done by their Scrum teams.

j) MetaScrum. It is a meeting attended by the Product Owner teams or their representatives and the stakeholders. The S@S framework (Sutherland 2019)

encourages to have this meeting as frequent as needed with once per Sprint as minimum. The goals of this meeting are getting the product backlog items ready to be implemented by addressing the needed strategy and resources.

k) Chief Product Owner (CPO). The CPO is part of a PO team and s/he is responsible for generating a single shared product backlog for all Scrum teams participating in the associated SoS. The CPO is further responsible of coordinating the priorities of the product backlog among the individual product owners of the individual Scrum teams. The role of the CPO is different from the role of the Scrum Mater of the Product Owner team and can be fulfilled by and an individual or by a group of Product Owners.

l) Executive MetaScrum (EMS). In S@S, a PO team is organically infinitely scalable, similarly to SoS. The PO team of the whole enterprise is called Executive MetaScrum (EMS). The EMS is the team responsible for establishing the vision and strategy of the entire enterprise together with the key stakeholders.

Table 3: Degree of agility of S@S.

S@S	Agility Features					
Phases	*FY*	*SD*	*LS*	*LG*	*RS*	*Total*
Pregame	0	0	0	1	0	1
Development	1	1	0	1	1	4
Postgame (Closure)	0	1	0	0	0	1
Total	**1**	**2**	**0**	**2**	**1**	**6**
Degree of Agility	***1/3***	***2/3***	***0/3***	***2/3***	***1/3***	***6/(3*5) = 0.4***
Practices						
SoS	1	1	0	1	1	4
Impediment removal backlog artefact	1	1	0	1	1	4
SoS backlog refinement meeting	1	1	0	1	1	4
Scaled Daily Scrum	1	1	0	1	1	4
SoS retro-spective	1	1	0	1	1	4
SoSM	1	1	0	1	1	4
SoSoS	1	1	0	1	1	4
EAT	1	1	0	1	1	4
PO team	1	1	0	1	1	4
MetaScrum	1	1	0	1	1	4
CPO	1	1	0	1	1	4
EMS	1	1	0	1	1	4
Total	**12**	**12**	**0**	**12**	**12**	**48**
Degree of Agility	***12/12***	***12/12***	***0/12***	***12/12***	***12/12***	***48/(12*5) = 0.8***

Following the analysis of S@S phases and practices, the first two authors checked whether a particular S@S practice supports the five agility features of the 4-DAT approach by separately answering the descriptive questions related to each agility feature in Table 1. If a S@S practice does support an agility feature, the score of 1 is assigned to that agility feature of that practice, otherwise the score of 0 is assigned. For example, the Pregame phase takes place only once in a Scrum project lifecycle, therefore changes in project risks or project vision are difficult to be accommodated – which made us assign 0 for FY (meaning that the phase does not support the agility feature FY). Scrum as well as S@S doesn't specify the maximum duration of the Pregame phase – which made us assign 0 for SD. The Pregame phase includes several important activities - such as defining and agreeing on a project vision, creating an initial architecture, creating scrum teams, which could use a lot of resources to implement them correctly – which means that we assign 0 for LS. Sharing knowledge and learning are increased when all people involved in the project discuss the project together – which means assigning 1 for LG. The Pregame includes a lot of important activities that occur once – which means assigning 0 for RS. After separately answering the descriptive questions and applying the scores, the first two authors came together and discussed the scores they separately have assigned to each agility feature of each particular S@S practice. Similar scores were confirmed, and different scores were resolved by conducting an argumentative discussion (Hitchcock 2002) between the two researchers to reach a shared rationally supported score. No unconfirmed scores remained after this argumentative discussion.

As Qumer and Henderson-Sellers (2008) suggest, for a method to have sufficient agility and be considered as an agile method, the calculated average of the degrees of agility on the four dimensions should be in the interval 0.5-0.6. However, the closer the calculated average is to 1, the higher the agility of the evaluated method. We have found as indicated in Table 3, that the average degree of agility of the S@S phases is 0,4, while it is 0,8 for the S@S practices. The total average degree of S@S (e.g. phases and practices) is therefore 0,6 which falls into the interval of 0.5-0.6 suggested by (Qumer and Henderson-Sellers, 2008) to consider a method as agile. Furthermore, the support of S@S practices for the agile values presented in Table 2, is also evaluated. The result of this evaluation is in Table 4. The second column of this table shows which S@S practice supports which agile value. As we can see, S@S has

several practices that explicitly support agile values, except the value "Keeping the process cost-effective" (see Table 4). We note that the S@S practices from the official literature, do not describe explicitly how one can keep the process cost-effective. We also note that in Table 3 we can clearly see also that none of the identified S@S practices support leanness which is another concept standing for cost-effectiveness of agile methods.

Table 4: The support of agility values.

Agile values	S@S Practices
Individuals and interactions over processes tools	SoS, SoS backlog refinement meeting, Scaled Daily Scrum, SoS retrospective, SoSM, SoSoS, EAT, Product Owner team, MetaScrum, EMS
Working software over comprehensive documentation	Game phase
Customer collaboration over contract negotiation	EMS, Pregame phase, Postgame phase
Responding to change over following a plan	Game phase, SoS backlog refinement meeting, MetaScrum, EMS
Keeping the process agile	Scaled Daily Scrum, SoS backlog refinement meeting, Impediment removal backlog artefact, SoS, SoSoS, Game phase
Keeping the process cost effective	-

4.2 Identifying S@S Practices Mitigating QRs Challenges

After evaluating the degree of agility of S@S, the first two authors have discussed and analyzed the identified S@S practices based on an argumentative discussion (Hitchcock 2002) to examine their fitness in mitigating the QRs challenges reported in (Alsaqaf, Daneva, and Wieringa, 2019) and reach a shared rationally supported mapping. The first two authors mapped therefore in an ongoing discussion the identified S@S practices to the reported categories of the challenges by using Conklin's dialog mapping technique for qualitative data structuring (Conklin, 2003). Table 5 summarizes this mapping. The first column of the table represents the reported categories and their related challenges, while the second column shows S@S practices that could be used to mitigate the related challenge in the first column. A dash "-" in the second column means that S@S does not explicitly specify a particular practice that could mitigate the reported QR challenge in the first column.

Table 5: Mapping S@S practices to QR challenges.

QR Challenges reported in (Alsaqaf et al. 2019)	S@S practices
Category 1: Teams coordination and communication challenges	
1.1.Late detection of QRs infeasibility	SoS, SoSoS, SoS backlog refinement meeting, Impediment removal backlog artefact, Scaled Daily Scrum
1.2.Hidden assumptions in inter-team collaboration.	SoS, SoSoS
1.3.Uneven teams maturity	EAT, SoS retrospective
1.4.Suboptimal inter-team organization	-
Category 2: Quality assurance challenges	
2.1.Inadequate QRs test specification	SoS, SoSoS, SoSM
2.2.Lack of cost-effective real integration test	-
2.3.Lengthy QRs acceptance checklist	-
2.4.Sporadic adherence to quality guidelines	Product Owner team, SoS, SoSoS
Category 3: QRs elicitation challenges	
3.1.Overlooking sources of QRs	CPO, Product Owner team, MetaScrum, EMS
3.2.Lack of QRs visibility	CPO, Product Owner team, MetaScrum, EMS, Pregame
3.3.Ambiguous QRs communication process.	CPO, SoS, SoSoS, Product Owner team, MetaScrum
Category 4: Conceptual challenges of QRs	
4.1.Unclear conceptual definition of QRs	-
4.2.Confusion about QR's specification approaches	CPO, MetaScrum, Product Owner team
Category 5: Architecture challenges	
5.1.Unmanaged architecture changes.	Impediment removal backlog artefact, SoS backlog refinement meeting, SoS, SoSoS, Product Owner team
5.2.Misunderstanding the architecture drivers	SoS, SoSoS, Product Owner team

S@S describes several practices that could (partially) mitigate one or more of the reported QRs challenges in (Alsaqaf et al., 2019) (see Table 5). For example, SoS teams could be used to establish clear communication channels among the distributed teams

with respect to QRs. Besides, the Product Owner team could shed light on the needed QRs based on their frequent communication with the stakeholders during the MetaScrums. Further, practices as SoS, SoSoS and PO team could help with setting up guidelines to distribute and share knowledge about internal quality aspects of the system (e.g. code style) which could result in satisfying internal quality such us maintainability and extendibility.

5 DISCUSSION

A Scrum team is "self-organizing" (Schwaber and Sutherland 2017), meaning that the Scrum team itself determines how to get the work done. However, in scaled agile, Scrum teams have to collaborate together to deliver customer's values. S@S moves the accountability for work coordination across Scrum teams, from the Scrum teams themselves to another team namely the Scrum of Scrum team. S@S uses Scrum of Scrum (SoS) – which is itself a Scrum team –to coordinate the work of multiple "self-organizing" Scrum teams. We were wondering if different "self-organizing" Scrum teams use different approaches to implement QRs. And if this is so, then how those teams will resolve inter-team conflicts? Mark Levison [4] - an agile practitioner - described the following example of a technical conflict between Scrum teams of one SoS: *"Given we're doing iterative development; teams are hopefully following the principles of emergent design. This means that we're writing high quality code, but not adding functionality or design structures until they are needed. Team A may write an encryptor without the use of an interface simply because they have need for only one. Team B may later need an encryptor which is slightly different from Team A's. What would be the best way for the organization to proceed is for team A to modify their code and have an encryptor interface - something that wasn't needed before. First, it's unlikely team B will even know about this. But if they do, Team A has no real incentive to help by modifying their code."* We think that this problem is caused by the fact that Scrum teams in S@S have no direct access to each other team's knowledge, since they have to communicate through an interface (e.g. SoS, SoSoS).

Table 5 indicates that we have not identified any S@S practices that could mitigate four reported QRs challenges (Alsaqaf et al., 2019) referring to Suboptimal inter-team organization, Lack of cost-effective real integration test, Lengthy QRs acceptance checklist and Unclear conceptual definition of QRs. S@S does not describe how to organize the Scrum teams around the product backlog items (e.g. component teams, feature teams). This issue was also reported as a problem by agile practitioners. For example, Mark Levison [5] has reported the following: *"Many scrum teams working together have serious problems delivering an end to end feature when several teams are involved. I have seen three separate teams, one comprised of UI people, one of mid-tier people and one of database people, get much more effective when they reorganized into three different teams organized around functionality. The people in the organized groups still performed more or less the same functions but were now able to swarm around features, not parts of a feature that was on a layer. This caused some integration problems across the teams but enabled end- to-end functionality to be built more quickly"*. Further, S@S describes the use of PO teams which are responsible for distilling the stakeholder's requirements, but doesn't mention explicitly how to treat the QRs or the customer's acceptance of those QRs. Moreover, SoS shifts the responsibility for delivering a fully integrated set of potentially shippable increments of product at the end of every Sprint from the Scrum team as described by the Scrum guide (Schwaber and Sutherland 2017) to the a SoS team which could resulted in cost-intensive process.

Table 3 indicates that S@S does not show leanness characteristics (column LS in Table 3). We do not claim that S@S phases or practices are not lean at all. We only demonstrate that those phases and practices are not compliant with the definition of lean as used in the 4-DAT tool, which we applied to analyze the agility characteristics of S@S. While the 4-DAT tool defines leanness in terms of waste reduction (see Table 1), S@S doesn't mention the concept of lean in its guide. However, not showing leanness does not reject the agility of a given method or framework, since leanness and agility have both different focus areas (Towill and Christopher, 2003). As per (Towill and Christopher, 2003), the lean approach is focused on eliminating waste and hence works well when the requirements are stable and predictable. Agile on the other side focuses more on increasing flexibility to deal with unpredictable and dynamic environments.

[4] https://www.infoq.com/news/2008/11/scrum-of-scrums

[5] https://www.infoq.com/news/2008/11/scrum-of-scrums

6 LIMITATION

We treated one specific framework (S@S), therefore, we cannot expect that the evaluation of the degree of agility would be representative for other scaled frameworks, e.g. SAFe and LeSS. This is a limitation. To counter it, we plan our next research step to be the application of the 4-DAT approach to evaluating the other frameworks included in our research (see Section 2.1). Furthermore, we treat the matching of S@S practices against the previously published QR challenges (Alsaqaf, Daneva, and Wieringa 2019) as a list of hypotheses (Wieringa and Daneva 2015) which we plan to explore in follow-up case studies. Currently, the first author is embedded in a large public organization that adopts large scale agile practices and in that we plan to carry out a multiple case study. This empirical research will include interviews and focus groups planned in multiple project teams.

Finally, qualitative research such as ours is always open to researchers' own bias. As the terms "quality requirements" and non-functional requirements" are not used in the S@S reference guide (Sutherland 2019), we had to use our own interpretation, experience and knowledge. However, we think that the possibility of misinterpretation is low, because both authors have a decade of experience in working with QRs, and the first author of the paper is a consultant and a certified Scrum master with industry experience in agile (so he has a sound professional understanding of the agile approaches as applied in practice). His interpretations during this research were grounded on his professional Scrum experience of using the Scrum terminology and definitions. Moreover, we countered the possible bias, by using Conklin's mapping technique consistently. Despite of this, we are considering important to further evaluate our mappings possibly with the participation of S@S experts from industry.

Last but not least, in using evaluation frameworks such as the 4-DAT analytical framework, there is always some risk of passing evaluator's bias. The 4-DAT framework evaluates agile methods from four perspectives and to counter the possibility of bias, the first two authors answered the descriptive questions of the 4-DAT analytical framework separately and thereafter based on an argumentative discussion (Hitchcock 2002) they discussed their answers to reach common supported judgment.

7 CONCLUSIONS AND IMPLICATIONS

This paper investigated the agile practices of the S@S framework from the perspective of QRs challenges identified in our earlier work (Alsaqaf, Daneva, and Wieringa 2019). We first assessed the degree of agility of S@S by using the 4-DAT approach. This indicated that the S@S supports the agile values defined by the Agile Manifesto (Agile Alliance 2001) (see Table 3 and Table 4), in the sense that it provides a scaling path to large and very large agile teams without deviating much from the agile philosophy due to incorporating heavyweight practices. We have then identified those S@S practices (see Table 5) that could be used to mitigate the QRs challenges reported in our previous work (Alsaqaf, Daneva, and Wieringa 2019). We found that S@S includes 12 practices that could (partially) mitigate one or more of the reported QRs challenges in (see Table 5). E.g., SoS teams could be used to establish clear communication channels among the distributed teams with respect to QRs. Besides, the PO team could shed light on the needed QRs based on their frequent communication with the stakeholders during the Meta Scrums.

However, our study found four QR challenges for which S@S offers no remedy. These are: Suboptimal inter-team organization, Lack of cost-effective real integration test, Lengthy QRs acceptance checklist and Unclear conceptual definition of QRs (Table 5). This has some practical implications. First, those practitioners conscious about QRs in projects that employ S@S, should take explicit actions towards creating practices that help counter these four challenges. E.g., practitioners should come up with their own ideas on how to manage the length of the QRs acceptance checklist, just because S@S offers no specific help in regard to this. On the other side, practitioners can rely on S@S in regard to coping with QRs challenges related to QRs elicitation and architecture. The design of S@S explicitly supports hierarchies of teams, empowerment and issue escalation processes, as well as the removal of roadblocks. This, in turn, is instrumental to the effective decision-making in resolving QRs issues.

Our immediate future work includes the evaluation of the degree of agility of the other scaled frameworks in our list and the matching of these frameworks' agile practices to the QR challenges identified in (Alsaqaf, Daneva, and Wieringa 2019).

REFERENCES

Agile Alliance. 2001. *Manifesto for Agile Software Development*. http://www.agilemanifesto.org.

Alsaqaf, Wasim, Maya Daneva, and Roel Wieringa. 2019. "Quality Requirements Challenges in the Context of Large-Scale Distributed Agile: An Empirical Study." *Information and Software Technology*.

Ambler, Scott W, and Mark Lines. 2012. *Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise*. IBM Press.

Appleton, J.V., Cowley, S. (1997). "Analysing Clinical Practice Guidelines. A Method of Documentary Analysis." *Journal of Advanced Nursing* 25(5): 1008–17.

Atkinson, P., Coffrey, A. (2004). *Analysing Documentary Realities. In D. Silverman (Ed.), Qualitative Research: Theory, Method and Practice (2nd Ed.)*. London, UK: Sage.

Beedle, M. (2018). "Enterprise Scrum Definition 4.0." http://www.enterprisescrum.com/.

Bick, S. et al. (2018). "Coordination Challenges in Large-Scale Software Development: A Case Study of Planning Misalignment in Hybrid Settings." *IEEE Transactions on Software Engineering* 44(10): 932–50.

Bowen, G. A. (2009). "Document Analysis as a Qualitative Research Method." *Qualitative Research Journal* 9(2): 27–40..

Calefato, F., Ebert, C. (2019). "Agile Collaboration for Distributed Teams." *IEEE Software* 36(1): 72–78..

COLLAB.NET, and VERSIONONE.COM. 2020. "14th Annual State of Agile Report." *VersionOne*. https://stateofagile.com/?_ga=2.145189495.27609247 1.1591726593-1008038165.1591726593#ufh-i-615706098-14th-annual-state-of-agile-report/7027494.

Conboy, K., Carroll, N. (2019). "Implementing Large-Scale Agile Frameworks: Challenges and Recommendations." *IEEE Software* 36(March/April): 1–9.

Conboy, K., Fitzgerald, B. (2004). "Toward a Conceptual Framework of Agile Methods: A Study of Agility in Different Disciplines." In *XP/Agile Universe 2004*, , pp 105-116.

Conklin, J. (2003). "Dialog Mapping: Reflections on an Industrial Strength Case Study." *Visualizing argumentation*: 1–15.

Hitchcock, D. (2002). "The Practice of Argumentative Discussion," Argumentation, vol. 16, no. 3, pp. 287–298.

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).

Kniberg, H., Ivarsson, A. (2012). *Scaling Agile @ Spotify - with Tribes, Squads, Chapters & Guilds*.

Krebs, J. (2008). *Agile Portfolio Management*. First Edit. Microsoft Press.

Larman, C., Vodde, B. (2016). *Large-Scale Scrum More with Less*. Pearson Education.

Leffingwell, D., Knaster, R. (2017). *SAFe 4.0 Distilled: Applying the Scaled Agile Framework for Lean Software and Systems Engineering*. 1st ed. Pearson Education.

Özcan-Top, Ö., Demirors, O. (2019). "Application of a Software Agility Assessment Model – AgilityMod in the Field." *Computer Standards and Interfaces* 62(July 2018): 1–16.

Portman, Henny. 2017. *Scaling Agile in Organisaties*. Van Haren Publ.

Putta, A., Paasivaara, M., Lassenius, C. (2018). "Benefits and Challenges of Adopting the Scaled Agile Framework (SAFe): Preliminary Results from a Multivocal Literature Review." In *PROFES 2018*, Springer International Publishing, 334–51.

Qumer, A., Henderson-Sellers, B. (2006). "Measuring Agility and Adaptibility of Agile Methods: A 4 Dimensional Analytical Tool." *Proceedings of the IADIS International Conference on Applied Computing* (January): 503–7. http://www.iadis.org.

Qumer, A., Henderson-Sellers, B. (2008). "An Evaluation of the Degree of Agility in Six Agile Methods and Its Applicability for Method Engineering." *Information and Software Technology* 50(4): 280–95.

Schwaber, K. (2018). "Nexus Guide - The Definitive Guide to Scaling Scrum with Nexus: The Rules of the Game." *Scrum.org* (January): 0–11. https://www.scrum.org/resources/nexus-guide.

Schwaber, K., Beedle, M. (2001). *Agile Software Development with Scrum*. First. Pearson..

Schwaber, K., Sutherland, J. (2017). "The Scrum Guide." *Scrum.Org and ScrumInc* (November): 19. http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-US.pdf.

Smart, J. (2018). "To Transform to Have Agility, Dont Do a Capital A, Capital T Agile Transformation." *IEEE Software* 35(6): 56–60.

Sutherland, J. (2001). "Agile Can Scale: Inventing and Reinventing SCRUM in Five Companies." *Cutter IT Journal* 14(12): 5–11.

———. 2019. "The Scrum@Scale Guide - The Definitive Guide to Scrum@Scale: Scaling That Works." *Scrum@Scale* (January): 1–19. https://www.scrumatscale.com/scrum-at-scale-guide/.

Towill, D., Christopher M. (2003). "The Supply Chain Strategy Conundrum: To Be Lean Or Agile or To Be Lean And Agile?" *International Journal of Logistics Research and Applications* 5(3): 299–309.

Wieringa, R.J., Daneva, M. (2015). Six strategies for generalizing software engineering theories. Sci. Comput. Program. 101: 136-152.