Analysing Large-Scale Scrum Practices with Respect to Quality Requirements Challenges

Wasim Alsaqaf[a], Maya Daneva[b] 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, Large-Scale Scrum, LeSS, Quality Requirements, Requirements Engineering, Non-functional Requirements, Documentary Research Method.

Abstract: Published empirical research using agile practitioners' perceptions indicated several important Quality Requirements (QRs) challenges experienced in agile large-scale distributed projects. It also indicated that a popular solution approach to those challenges is to inject some heavyweight practices into agile, for example adding documentation or roles of authorities for QRs. At the same time, agile methodologists proposed several scaled agile frameworks to specifically serve agile organizations working on large and distributed projects. How do these frameworks address QRs? Do they put forward any heavyweight practices as a solution to QRs challenges, or do they invent new agile practices fully aligned with the values of the Agile Manifesto? Currently, very little is known about the extent to which the QRs issues are accounted for in the design of these frameworks as proposed by agile methodologists. This paper attempts to narrow this gap of knowledge. It analyses Large-Scale Scrum (LeSS), a prominent scaled framework, from the perspective of QRs engineering and the Agile Manifesto's values. To this end, we first applied the 4-Dimensional Analytical Tool to evaluate the degree of agility of the practices of the LeSS framework. We then analysed these practices and evaluated their applicability to mitigate the QRs challenges reported in previous work.

1 INTRODUCTION

The urgent need of organizations to react quickly to the rapidly changing environment forces them to embrace agility and get rid of the traditional heavyweight development methodologies (Helmy et al., 2012). The 13th annual state-of-agile report (Collab.net & Versionone.com, 2019) stated the following reasons for adopting agile in general:
1) Accelerate software delivery, 2) Enhance ability to manage changing priorities, 3) Increase productivity, 4) Improve business/IT alignment, 5) Enhance software quality, 6) Enhance delivery predictability, 7) Improve project visibility, 8) Reduce project cost, 9) Improve team morale, 10) Reduce project risk, 11) Improve engineering discipline, 12) Increase software maintainability, 13) Better manage distributed teams. The adoption of agile is driven by the benefits companies strive to achieve (e.g. Ability to manage changing priorities, Project visibility, Business/IT alignment, Delivery speed/time to market) (Collab.net & Versionone.com, 2019). However, moving from the original context for which agile methods were designed for - small single, co-located teams – toward the implementation of agile development methods in large-scaled distributed context is not a flawless transformation (e.g. communications challenges, coordination challenges, lack of flexibility) (Conboy & Carroll, 2019).

To provide guidelines on how to address the challenges of the transformation to large-scaled distributed agile, many agile scaled frameworks such as Scaled Agile Framework (SAFe) (Leffingwell & Knaster, 2017), Large-Scale Scrum (LeSS) (Larman & Vodde, 2016), Spotify (Kniberg & Ivarsson, 2012), Scrum of Scrums (SoS) (Sutherland, 2001) have been introduced and used since the creation of the Agile Manifesto (Agile Alliance, 2001). Based on the principles of the Manifesto, each framework defines its own structure to guide practitioners through scaling agile. However, there is scarcity of evidence regarding the evaluation of agility's degree of these

[a] https://orcid.org/0000-0002-0253-428X
[b] https://orcid.org/0000-0001-7359-8013

Alsaqaf, W., Daneva, M. and Wieringa, R.
Analysing Large-Scale Scrum Practices with Respect to Quality Requirements Challenges.
DOI: 10.5220/0009680600920103
In *Proceedings of the 15th International Conference on Software Technologies (ICSOFT 2020)*, pages 92-103
ISBN: 978-989-758-443-5

agile scaled frameworks (Qumer & Henderson-Sellers, 2008) and the use and effectiveness of the agile scaled frameworks in real life projects (Conboy & Carroll, 2019). Moreover, in our previous empirical research (Alsaqaf et al., 2019) we have investigated one particular aspect of scaling agile, namely the engineering of Quality Requirements (QRs) such as security, performance, usability, availability. Based on an exploratory case study on how agile practitioners treat QR's, we concluded that engineering QRs remains a challenge in agile context (be it small single co-located or large-scaled distributed). Particularly in large-scaled projects, our study has further identified 15 QRs challenges, 13 mechanisms behind the challenges and 9 practices agile practitioners currently use to cope with the identified challenges. 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 mitigating the QRs challenges which were identified in our previous work (Alsaqaf et al., 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. We make the note that our previously published empirical work (Alsaqaf et al., 2019) revealed a tendency of many large agile project organizations to counter their QRs challenges by introducing heavyweight practices that add new artefacts (e.g. security stories), roles (e.g. security officer) and activities to the agile process. In turn, adding such practices often means making the process less agile (Alsaqaf et al., 2019).

The present paper reports our results of analysing one specific scaled framework, namely, LeSS (Larman & Vodde, 2016). Our ongoing research also includes some other scaled frameworks, however these are out of scope in this paper. To this end, we set out to answer the following research question: *RQ: What are the agile practices suggested by the LeSS agile scaled framework that could mitigate the effect of the QRs challenges identified in (Alsaqaf et al., 2019)?* Using a documentary research process, we analysed the practices that the LeSS methodologists (Larman & Vodde, 2016) proposed to use in large projects including of up to 8 teams (LeSS) and projects with more than 8 teams (LeSS Huge).

The rest of this paper is structured as follow: Section 2 describes our research process. Section 3 provides some background and definitions of the most important concepts. Section 4 presents our results. Section 5 discusses them. Section 6 is on limitations of our research and Section 7 is on implications.

2 RESEARCH PROCESS

The overall goal 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 (Alsaqaf et al., 2019). To achieve this goal, we used the following steps inspired by documentary research methods (Appleton & Cowley, 1997), as depicted in Figure 1.

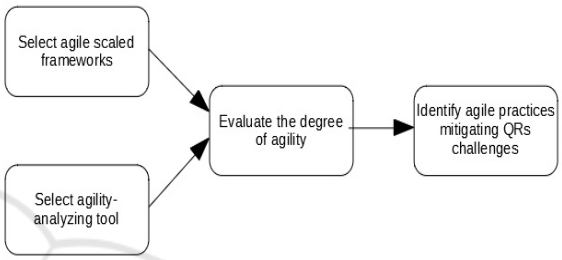

Figure 1: Our research Process.

Our process includes: (1) selecting agile scaled frameworks for inclusion in the research, (2) evaluating the degree of agility of their proposed practices, and (3) evaluating the extent to which the practices proposed in the frameworks mitigate the QRs challenges identified in (Alsaqaf et al., 2019). Step (1) explains our reasoning for including certain frameworks. Step (2) is concerned with the evaluation of how much agile the practices of a scaled framework are as is described by the authors of the framework in their framework's documentation (and not as implemented in a particular organization). Step (3) is concerned with the matching of the agile practices proposed by the authors of the scaled framework against the QRs challenges found in (Alsaqaf et al., 2019). As this paper is focused on one framework only LeSS (Larman & Vodde, 2016), 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. He classified these into two categories, namely i) Enterprise-targeted frameworks (e.g. SAFe (Leffingwell & Knaster, 2017), LeSS (Larman & Vodde, 2016), Nexus

(Schwaber, 2018), Scrum@Scale (S@S) (Sutherland, 2019)) which are used to deliver complex enterprise-level products whereby the collaboration between distributed teams is essential and ii) Web scale-targeted frameworks (e.g. Spotify (Kniberg & Ivarsson, 2012), Scaled Agile Lean Development (http://scaledprinciples.org/)) which are used to support the IT-department of an organization 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-scaled context. Furthermore, we limit our selection of frameworks to those that are the most used according to (Collab.net & Versionone.com, 2019). The 13th annual state-of-agile report (Collab.net & Versionone.com, 2019) has indicated the following agile scaled frameworks as the most popular:(i) SAFe (Leffingwell & Knaster, 2017), (ii) SoS (Sutherland, 2001), (iii) Internally created methods, (iv) Disciplined Agile Delivery (DAD) (Ambler & Lines, 2012), (v) Spotify (Kniberg & Ivarsson, 2012), (vi) LeSS (Larman & Vodde, 2016), (vii) Enterprise Scrum (ES) (Beedle, 2018), (viii) Lean management (https://www.lean.org/), (ix) Agile Portfolio Management (AgilePM) (Krebs, 2008), (x) Nexus (Schwaber, 2018), (xi) Recipes for Agile Governance in the Enterprise (RAGE) (https://www.cprime.com/rage/). The intersection between the Enterprise-targeted frameworks in (Portman, 2017) and the most popular agile scaled framework described in (Collab.net & Versionone.com, 2019) reduces our selection group to SAFe, LeSS, Nexus, S@S, SoS, DAD, ES, AgilePM, Lean management and RAGE. In this paper we chose to focus on the agile practices of the LeSS framework, because of the simplified organizational design it introduces, and because it is less process-heavy (compared to e.g. SAFe). Besides, LeSS is grounded on Scrum (Schwaber & Sutherland, 2016) which is the most used agile method (Collab.net & Versionone.com, 2019). However, our choice for LeSS does not mean that we prefer or recommend LeSS. The other frameworks will be investigated in our follow-up research.

2.2 Evaluating the Degree of Agility

Since the introduction of the Agile Manifesto, 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 (Agile Alliance, 2001). However, while creating a framework for scaling up agile, it might well be possible that the framework's authors introduce some heavyweight practices into it. This is because scaling up agile necessarily involves some balancing of agility, discipline, organizational structures, coordination mechanisms and roles (Conboy & Carroll, 2019). In fact, a 2018 literature review (Abheeshta et al., 2018) on the adoption of the SAFe framework reports the "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 the 4-Dimensional Analytical Tool (4-DAT) described by (Qumer & Henderson-Sellers, 2006) in order to evaluate the degree of agility of LeSS (Larman & Vodde, 2016). We note that there are other approaches that assess the agility factor of an agile software development framework such as the Conceptual Framework of Agile Methods described by (Conboy & Fitzgerald, 2004) and the AgilityMod approach of (Özcan-Top & Demirors, 2019). However, in contrast to the 4-DAT approach (Qumer & Henderson-Sellers, 2006) which is focused on the agile practices of the agile scaled framework itself, these other assessment frameworks (Conboy & Fitzgerald, 2004) and (Özcan-Top & Demirors, 2019) focus on the agility factor of the particular organizational application of the particular framework's practices by agile teams. We 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 (Conboy & Carroll, 2019)). 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 scaled framework. Taking into account that the 13th annual state of agile report (Collab.net & Versionone.com, 2019) has stated among others: Lack of skills/experience with agile methods, Insufficient training and education, and Inconsistent processes and practices across teams, as challenges experienced in scaling agile, we decided to evaluate the practices of LeSS as described by its literature (Larman & Vodde, 2016).

2.3 Identifying Practices Mitigating QRs Challenges

The literature provided by LeSS (Larman & Vodde, 2016) in its official website (www.less.works) was investigated. The suggested practices were analysed based on their description and fitness to mitigate the QRs challenges identified in (Alsaqaf et al., 2019). The analysis started with reading and re-reading the reference document of LeSS and the information on the www.less.works site that pertains to the 27 large companies that implemented LeSS, which served as input. The first two authors then checked the relevance of each LeSS practice for mitigating the QRs challenges identified in (Alsaqaf et al., 2019). For clarity, we list these challenges as follows: i) Late detection of QRs infeasibility: QRs are not visible in the same way Functional Requirements (FRs) do and, in turn, identifying and designing acceptance tests for them may be difficult, too. ii) Assumptions in inter-team collaboration: Teams make tacit assumptions about inter-team collaboration. iii) Uneven teams' maturity: Team maturity levels across teams vary, some teams are more mature than others and act differently on QRs. iv) Suboptimal inter-team organization: Large agile projects organize agile teams differently around the Product backlogs (e.g. component teams, Feature teams). A suboptimal teams' organization choice could affect negatively the quality attributes of the system. v) Inadequate QRs test specification: Agile teams report the complexity of modelling QR's and therefore identifying and designing acceptance tests for them may be difficult. vi) Simulated integration tests: Agile teams simulate integrated tests due to the lack of cost-effective real integration test for QRs, which results in identifying QRs defects late in the development cycle. vii) End user acceptance of QRs: Agile teams in some occasions use the so-called 'definition-of-done' (DoD) to specify the conditions for deeming the QRs 'met' by the delivered product, which could result in a very long DoD. viii) Strict adherence to quality guidelines: Coordination on standards and compliance is hard. ix) Overlooking sources of QRs: Lack of an adequate process to identify the right stakeholders, which could result in missing QRs or not identifying them at the right time. x) Lack of QRs visibility: Hard to keep attention on internal qualities that matter to developers e.g. maintainability, modifiability, extensibility. xi) Ambiguous QRs communication process: who owns the QR's and how to describe them. xii) Conceptual definition of QRs: Lack of clarity on how to treat QRs – are FRs and QRs to be treated the same way? Or QRs are 'standalone' requirements and need separate treatment? xiii) Mixed specification approaches to QRs: Confusion about QR's specification approaches. How to record them? xiv) Unmanaged architecture changes: Changes made to QRs at any time in the development cycle could result in costly changes in the software architecture because the earlier architecture becomes inappropriate for the new QRs. xv) Misunderstanding the architecture drivers. Conflicting ideas of which QRs drive the architecture and why architecture trade-offs are made in a particular way.

3 BACKGROUND AND DEFINITIONS

3.1 LeSS

Large-Scale Scrum (LeSS) is developed by two agile practitioners: Larman (www.craiglarman.com) and Vodde (www.odd-e.com). It is built upon the fundamentals of Scrum. The authors (Larman & Vodde, 2016) describe LeSS as being "Scrum applied to many teams working together on one product". Based on this description, LeSS includes all those roles, events and artifacts that make up Scrum. However, LeSS differs from Scrum in the following: i) LeSS proposes the so-called 'Sprint planning one', it is a meeting for all teams to decide which team will work on which part of the product backlog, ii) Daily scrum, it is an event known from Scrum, though LeSS explicitly encourages teams to visit each other's daily scrum for observation to increase knowledge sharing. iii) Coordination and integration, LeSS supports decentralized team's coordination iv) Overall product backlog refinement, it is a meeting attended by the product owner (PO) and representatives of all teams to decide which items are likely to be implemented in the next sprint and select them for further single-team product backlog refinement v) Product backlog refinement, it is the same single-team refinement session as described by Scrum, however LeSS encourages that multiple teams perform this session together to increase learning and help coordination. vi) Sprint review, in comparison with Scrum, LeSS uses different techniques to conduct this event (e.g. diverge-converge meeting pattern). vii) Overall retrospective, it is a LeSS meeting for approximately 45 minutes at the end of each sprint where teams' representatives, PO and scrum masters explore improving the whole development process.

Furthermore, LeSS includes: i) Rules that define

the key structures of LeSS and have to be implemented as part of the application of LeSS, ii) Principles that shape the philosophy behind LeSS and need to be considered when applying LeSS in a company's own context, iii) Guides which are a set of tips to help practitioners adopting the LeSS rules and iv) the so-called Experiments which provide some examples of applying LeSS.

There are two variants of LeSS: i) LeSS for 2 to 8 teams, and ii) LeSS Huge, meant for more than 8 teams. Both variants postulate that there is one PO and one product backlog for the whole product, one common 'definition of done' shared by all teams, one common sprint across all teams and one shippable product increment delivered at the end of each sprint.

LeSS Huge consists of multiple implementation of LeSS and is meant to help very large agile organizations (with projects including more than 8 teams) achieving the value and simplicity of Scrum while introducing as few additional concepts as possible to the project organization. These concepts are the Requirements Area (an attribute of the product backlog that allows to view all those items specific to an area of importance to stakeholders), the Area Product Owners (forming one team with the PO), the Area Feature Teams (responsible for implementing those items in the product backlog belonging to a particular area).

3.2 4-Dimensional Analytical Tool (4-DAT)

This section explains the evaluation model that we use for understanding the degree of agility of LeSS. Qumer and Henderson-Sellers (2006) have developed the 4-DAT tool to compare agile methods and evaluate their degree of agility. The 4-DAT tool analyses an agile method in terms of four dimensions and is extendable with additional dimensions as needed.

3.2.1 Dimension 1 – Method Scope Characterizations

This dimension 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).

3.2.2 Dimension 2 – Agility Characterizations

The second dimension is a set of agility features to measure the agility of a given method. The agility features are: flexibility (FY), speed (SD), leanness (LS), learning (LG) and responsiveness (RS) (see Table 1). The authors derived these agility features from the following definition of agility they have offered based on assessing existing definitions 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.'' (Qumer & Henderson-Sellers, 2006).

Table 1: Agility features and description.

Agility feature	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?

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, only 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 & Henderson-Sellers, 2008).

For a method to have sufficient agility and be considered as agile method, the calculated average of the degrees of agility 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. Table 2 gives examples - as provided by (Qumer &

Henderson-Sellers, 2008) - of measuring the agility degree of Scrum's phases as representative of agile methods. Table 2 shows that the authors recognise speed in all Scrum's phases and therefore assign 1 for speed. In contrast the authors report the lack of leanness in all Scrum's phases.

Table 2: Degree of Agility of Scrum.

Scrum	Agility Features					
Phases	FY	SD	LS	LG	RS	Total
Pre-Game	1	1	0	1	1	4
Development	1	1	0	1	1	4
Post-Game	0	1	0	0	0	1
Total	2	3	0	2	2	9
Degree of Agility	2/3	3/3	0/3	2/3	2/3	9/(3*5) = 0.6

3.2.3 Dimension 3 – Agile Values Characterizations

Dimension 3 evaluates whether the practices of the agile method to be examined supports six agile values. Four of those are the values provided by the Agile Manifesto. The other two were reported by (Qumer & Henderson-Sellers, 2006) (see Table 3).

Table 3: Agile values and description.

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?

3.2.4 Dimension 4 – Software Process Characterization

This dimension examines those practices of the agile method that support four components of the software development process, namely: i) Development process, ii) Project management process, iii) Support process, and iv) Process management process.

3.2.5 Application of 4-DAT in Our Research

Based on the 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.

In the next section, we first describe the practices and phases of LeSS and LeSS Huge that were subjected to our evaluation on Dimensions 2 and 3, and then present how these practices possibly mitigate those QRs challenges identified in (Alsaqaf et al., 2019).

4 FINDINGS

4.1 Evaluating the Degree of Agility of Less

The LeSS framework was analysed to explore LeSS phases and practices and to measure their agility in terms of the aforementioned agility features (see Table 1).

4.1.1 Less Practices

As already noted, LeSS is built upon Scrum (Schwaber & Sutherland, 2016) and hence shares Scrum roles, artefacts and practices such as scrum master, PO, product backlog, sprint planning (in LeSS terminology called 'sprint planning two'), single-team product backlog refinement, single-team retrospective and sprint. Below, we discuss and analyse only the LeSS practices that are different in their implementation from those in Scrum or are newly introduced in LeSS (see subsections a-k, below) and LeSS Huge (see l-n).

a) Initial Product Backlog Refinement (Initial-PBR) is a one-off not repeatable meeting which is conducted at the very beginning of the project. During the Initial-PBR many important activities take place e.g. creating and filling the product backlog, identifying risks, defining project vision. The Initial-PBR meeting is held at one place and meant to establish shared knowledge among all people involved in carrying out the project within at least two days.
b) Sprint planning one. This is a meeting attended by the PO and all teams' members or their representatives. Its goals are to tackle open questions, to fill each team's sprint backlog with

the product backlogs items which each team has selected to work on, to define the sprint goal and to identify possible inter-team collaborations.

c) Multi-team Sprint planning two. This is done by having two or more teams meet in the same physical location with each team conducting its own single-team sprint planning two meeting (analogous to scrum sprint planning). Multi-team sprint planning two is used when two or more teams have to implement features that affect the same part of the system. The teams in that particular situation need to collaborate to resolve design issues and coordinate shared work. During a multi-team sprint planning two, a multi-team design workshop session can be conducted to tailor out the design of complex shared items. Besides, test scenarios can be structured during multi-team sprint planning two to make the development progress visible to the stakeholders during the sprint review.

d) Overall product backlog refinement. It is a product backlog refinement meeting shared among all teams together with the PO. This meeting holds before the single-team product backlog refinement meeting. The goals of this meeting are to assign product backlog items to the teams, to split big items, to estimate items and to identify team's collaboration possibilities.

e) Multi-team product backlog refinement. It is a product backlog refinement meeting shared amongst two or more teams working on related features to coordinate their work.

f) Sprint review. LeSS suggests the use of diverge-converge meetings to conduct the sprint review at the end of each LeSS sprint. During diverge periods, a bazar is established where all teams or their representatives demonstrate their implemented items. The stakeholders visit the team's area of their interest and discuss the delivered work. In the converge periods, the stakeholders summarize what they have seen and inspect the items of interest. Multiple diverge-converge meeting cycles could be conducted if needed.

g) Overall retrospective meeting. This meeting occurs right after the team's retrospective and is attended by all teams, PO and scrum masters to discuss cross-team, organizational and systemic problems within the organization. LeSS suggests to use Cause-Effect diagrams during the overall retrospective to investigate problems and possible solutions.

h) Daily Scrum. The LeSS daily scrum is identical with the one described in Scrum, however LeSS encourages teams's representatives to attend the daily scrum of the other teams to enhance knowledge sharing.

i) LeSS teams. While Scrum advocates self-organizing teams, LeSS goes beyond that by emphasizing the need for self-managing teams. A self-managing LeSS team has the authority to design, implement and monitor their work. They take all the responsibility of how to get the work done. Moreover, LeSS encourages the use of features teams above component teams. A feature team is a team that has all the skills needed to implement a customer value across components. Members of the feature teams must be fully dedicated to one team, namely their own team to enhance sharing responsibility of the team's goal. Further, LeSS suggests the use of leading teams to guide the implementation of large features. Large features have to be split up into small product backlog items. These small items would be allotted to different feature teams to be implemented. The leading team of a large feature would then coordinate the work among the feature teams to ensure the right implementation of that particular large feature.

j) Communities of Practice (CoP). These are voluntary groups of people who share the same interest. Each CoP is used to enhance the knowledge of the participants on particular subject. It is a self-organized group with no decision rights.

k) Definition of Done (DoD). LeSS encourages the use of the DoD, a practice well-known from Scrum. Moreover, LeSS emphasizes that one common definition of done must exist for the whole product and must be shared by all involved teams. Each team may however strengthen the common definition of done by expanding it. A perfect DoD from LeSS' perspective is the one that includes all needed activities to ship the whole product at the end of each sprint to the end users with the new "completed" and "done" features. LeSS team should improve the initial DoD frequently toward the perfect one. Furthermore, LeSS makes a clear distinction between the DoD as described above and the acceptance criteria which is a list of specific stipulations demanded for each particular product backlog item to be accepted by the stakeholders.

l) Requirements area. This is part of the LeSS Huge implementation. It is a customer-centric category of the requirements. For example, as provided by (Larman & Vodde, 2016), for a telecom system, some of the requirements areas could be

Protocols, Performance, and Network Management.

m) Area product backlog. This is part of LeSS Huge and includes those product backlog items that relate to one requirements area. These items are grouped into one area product backlog. An area product backlog is therefore only part of the whole product backlog.
n) Area product owner. While the PO in both LeSS and LeSS Huge is responsible for the whole product backlog, an area PO is the one responsible for a particular area product backlog in LeSS Huge. (S)he is specialized in the requirement area of that area product backlog and acts as the PO of that area to the feature teams involved in implementing it.

4.1.2 Less Phases

The literature of LeSS (i.e. the official website www.less.works) and (Larman & Vodde, 2016)) do not mention any phases in the LeSS framework. However, since LeSS is built upon the fundamentals of Scrum, we assume that the phases of Scrum as described in (Schwaber & Beedle, 2001) are applied to LeSS. We chose further not to investigate the Scrum phases and check their agility factor because it is beyond the scope of this paper where the focus is put on the LeSS framework. Besides, the agility of the Scrum phases was analysed in (Qumer & Henderson-Sellers, 2008) (see Table 2).

Following the analysis of LeSS phases and practices, the first two researchers checked whether a particular LeSS 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 LeSS practice does support an agility feature, the score of 1 is assigned to that agility feature of that practice, otherwise a 0 is assigned. Thereafter, the researchers came together and discussed the scores they separately have applied to each agility feature of each particular LeSS 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.

Table 4 shows the support of the LeSS practices for the agility features as described in Table 1. For example, the Initial-PBR takes place only once in a LeSS project lifecycle, therefore changes in project risks or project vision are difficult to be accommodated – which lets us assign 0 for FY (meaning that the phase does not support the agility feature FY). The Initial-PBR could last for two or more days, LeSS however did not specify the maximum duration of this meeting – which lets us assign 0 for SD. The Initial-PBR requires all people involved in the project (e.g. scrum masters, team members, stakeholders, PO) to attend the meeting – 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 – 1 for LG. The Initial-PBR requires a lot of people at one place to discuss once different aspects of the project – 0 for RS.

Table 4: Degree of Agility of LeSS.

LeSS	Agility Features					
Practices	FY	SD	LS	LG	RS	Total
Initial-PBR	0	0	0	1	0	1
Sprint planning one	1	1	0	1	1	4
Multi-team sprint planning two	1	1	0	1	1	4
Multi-team product backlog refinement	1	1	0	1	1	4
Overall product backlog refinement	1	1	0	1	1	4
Sprint review	0	0	0	1	0	1
Overall retrospective meeting	1	1	0	1	1	4
Daily scrum	1	1	0	1	1	4
LeSS teams	1	1	0	1	1	4
Communities of Practice	1	1	0	1	1	4
Definition of Done	1	1	0	1	1	4
Requirements area	1	1	0	1	1	4
Area product backlog	1	1	0	1	1	4
Area PO	1	1	0	1	1	4
Total	12	12	0	13	12	50
Degree of Agility	12/14	12/14	0/14	14/14	12/14	50/(14*5) = 0.72

Furthermore, the support of LeSS practices for the agile values presented in Table 3, is also evaluated.

We note that LeSS has several practices that explicitly support agile values except the value "Keeping the process cost effective" (see Table 5). The identified LeSS practices do not describe explicitly how to keep the process cost effective. In Table 4 we can clearly see that none of the identified LeSS practices support Leanness. We think that the lack of support for Lean shown by LeSS practices is in fact caused by the different Lean definitions utilized in 4-DAT tool and the LeSS framework.

Table 5: The support of Agility values.

Agile Values	LeSS practices
Individuals and interactions over processes tools	Sprint planning one, Sprint planning two, Multi-team sprint planning two, Product backlog refinement, Multi-team product backlog refinement, Overall product backlog refinement, Sprint review, Team retrospective, Overall retrospective meeting, Daily scrum, LeSS teams
Working software over comprehensive documentation	Sprint planning two, Sprint, Sprint review
Customer collaboration over contract negotiation	Initial-PBR, PO, Requirements area
Responding to change over following a plan	Sprint review, Sprint planning one, Sprint planning two, Multi-team sprint planning two
Keeping the process agile	Daily scrum, Sprint review, Team retrospective, Overall retrospective meeting, LeSS teams.
Keeping the process cost effective	-

4.2 Identifying LeSS Practices Mitigating QRs Challenges

After evaluating the degree of agility of LeSS, we have analysed the identified LeSS practices to examine their fitness in mitigating the QRs challenges reported in (Alsaqaf et al., 2019). We mapped therefore the identified LeSS practices to the reported categories of the challenges by using Conklin's dialog mapping technique for qualitative data structuring (Conklin, 2003). Table 6 summarizes this mapping. The first column of the table represents the reported challenges (see Section 2.3), while the second column shows LeSS practices that could be used to mitigate the related challenge in the first column. A dash "-" in the second column means that LeSS does not explicitly specify a particular practice that could mitigate the reported QR challenge in the first column.

Table 6: Mapping LeSS practices to challenges.

Challenges	LeSS practices
Late detection of QRs infeasibility	Sprint planning one; Multi-team sprint planning two, Multi-team product backlog refinement, Overall product backlog refinement, Daily scrum
Assumptions in inter-team collaboration	LeSS teams
Uneven teams maturity	Communities of Practice
Suboptimal inter-team organization	LeSS teams
Inadequate QRs test specification	Sprint planning one; (Multi-team) sprint planning two, definition of done
Simulated integration tests	Definition of Done
End user acceptance of QRs	-
Strict adherence to quality guidelines	LeSS teams, Multi-team planning two, Definition of Done
Overlooking sources of QRs	Initial-PBR, Sprint review
Lack of QRs visibility	Requirements area, Area product backlog, Area PO
Ambiguous QRs communication process	Requirements area, Area product backlog, Area PO
Conceptual definition of QRs	-
Mixed specification approaches to QRs	Requirements area, Area product backlog, Area PO, Communities of Practice
Unmanaged architecture changes	Multi-team sprint planning two
Misunderstanding the architecture drivers	Initial-PBR, Multi-team sprint planning two

LeSS recognizes several practices that could (partially) mitigate one or more of the reported QRs challenges in (Alsaqaf et al., 2019) (see Table 6). For example, Communities of practice could be used to establish cross-team discussions of quality assurance related subjects (e.g. automation tests, test driven development, quality standard definitions). Besides, establishing communities of practices that discuss a particular QR (e.g. performance, usability, security) or the concept of QR as whole could help mitigating the conceptual challenges of QRs.

5 DISCUSSION

This section discusses first the evaluation of the LeSS' degree of agility and then our reflection on LeSS deals with the QRs challenges identified in (Alsaqaf et al., 2019).

Table 4 indicates that LeSS does not show lean characteristics. We do not claim that LeSS 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 applied to analyse the agility characteristics of LeSS. While the 4-DAT tool defines leanness in terms of waste reduction (see Table 1), LeSS emphasizes that 'lean' is not waste reduction but it is about showing respect to people and continuous improvement. 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 & Christopher, 2003): the lean approach is focused on eliminating waste and hence works well when the requirements are stable and predictable; agile, in the other side, focuses more on increasing flexibility to deal with unpredictable and dynamic environments.

LeSS emphasizes that a team member should be dedicated 100% of the time to one team. It means that periodically shuffling of people between the LeSS teams is not encouraged. LeSS recognized this demand as inflexible, however, LeSS considers sharing responsibility of the team's goal very important. We are wondering if keeping team members 100% assigned to one team only translates in creating silos where the responsibility for the team's goal gets higher priority than the responsibility for the whole product, especially when allocating teams to particular requirements area?

We have not identified any LeSS practices that could mitigate the QR challenges: *End user acceptance of QRs* and *Conceptual definition of QRs*. LeSS makes a clear distinction between i) a DoD that includes all activities needed to ship the whole product to the end users, and ii) the acceptance criteria which is a list of conditions that must be met by each product backlog item to be accepted by the stakeholders. LeSS does not provide guidelines about how to specify the acceptance criteria, while the steps needed to structure the DoD are very explicit (Larman & Vodde, 2016). Further, LeSS recognizes that agile practitioners often neglect QRs because they believe that QRs cannot be specified and tested. Therefore, LeSS encourages agile practitioners to treat QRs as functional requirements and does not provide further any guidelines about how to handle the QRs specifically.

In Table 6, three LeSS practices, namely, requirements area, area product backlog and area PO were identified as possible mitigations to the QRs challenges *Lack of QRs visibility*, *Ambiguous QRs communication process* and *Mixed specification approaches to QRs* However, it turns out that these LeSS practices are only available when applying LeSS Huge (Larman & Vodde, 2016). Hence, for projects with less than nine teams, they have fewer LeSS practices available to deal with the QRs elicitation challenges and the conceptual challenges of QRs namely, only sprint review and communities of practice.

We have observed that LeSS uses the term 'self-managing' when it describes the responsibilities of LeSS teams. 'Self-managing' is one of the four types of teams described in LeSS (Larman & Vodde, 2016), namely: i) Manager-lead teams, ii) Self-managing teams, iii) Self-designing teams and iv) Self-governing teams. LeSS defines the 'self-managing' team as "The team'[that] is responsible for executing the tasks and monitoring and managing process and progress". Scrum (Schwaber & Sutherland, 2016) however uses the term 'self-organizing' to describe Scrum teams, which is not part of the four types of teams described in LeSS. Agile practitioners seem to have different interpretations of 'self-managing' team as described by LeSS versus 'self-organizing' as described by Scrum. For example, Mario Moreira, an agile practitioner, provided in his blog (http://cmforagile.blogspot.com/2017/07/what-is-self-management-and-is-it-good.html) different definitions for 'self-managing' and 'self-organizing' where 'self-managing' teams in his opinion have more authority than 'self-organizing' teams. It is important to explicitly describe the responsibilities of an agile team to avoid any assumptions that could resulted in confusions by the teams regarding the right implementation of the requirements in general and QRs in specific.

In Table 6, we have identified Initial-PBR and Sprint review as possible mitigations to the QRs challenge *Overlooking sources of QRs*. These two LeSS practices however have the lowest degree of agility in comparison with other LeSS practices (see table 4). This observation could mean that heavyweight practices need to be injected into agile scaled frameworks to cope with QRs challenges, which is in line with the reported findings of previous studies (e.g. Abheeshta et al., 2018, West et al., 2011).

LeSS explicitly moves the responsibility of inter-teamwork coordination to the teams themselves. We observe this in those LeSS practices that require team coordination (e.g. multi-team sprint planning two,

multi-team product backlog refinement, overall product backlog refinement, overall retrospective meeting). Giving the coordination's responsibility to the involved teams could mitigate the team's coordination and communication challenges reported in (Alsaqaf et al., 2019), if the teams are mature enough, and in turn know how to act. We think that the lack of team's maturity could possibly result in abusing the given responsibility and lead to chaos.

6 LIMITATIONS

This study dealt with one specific framework, namely LeSS. We therefore cannot expect that the evaluation of the degree of agility would be similar for other scaled frameworks, e.g. SAFe and Scrum of Scrums. This is a limitation. In line with this, our immediate plan and future work is to apply the 4-DAT approach to evaluating the other frameworks included in our research (see Section 2.1).

Moreover, this research is based on the investigation of the literature provided by LeSS (Larman & Vodde, 2016) in its official website (www.less.works) and represents the ideas of the authors and their interpretation of the LeSS literature. We acknowledge that the presented results might possibly be different if external LeSS practitioners have been involved in this research. Therefore, as part of our immediate future work, we plan to evaluate the results of this research by interviewing LeSS experts in the Netherlands and discuss the results with them.

Furthermore, the matching of LeSS and LeSS Huge practices against the previously published challenges (Alsaqaf et al., 2019) is a list of hypotheses at best. As the terms "quality requirements" and non-functional requirements" are not used in the LeSS reference book (Larman & Vodde, 2016), we had to use our own interpretation, experience and knowledge. This could be partly subjective. However, we countered this issue by using Conklin's mapping technique consistently. Despite of this, we are considering important to further evaluate our mappings possibly with the participation of LeSS experts from industry.

7 CONCLUSION

This paper investigated how QRs issues that were identified in prior research, are treated in LeSS (Larman & Vodde, 2016), an agile scaled framework. We first assessed the degree of agility of LeSS by using the 4-dimensional analytical tool 4-DAT. This assessment indicated that the LeSS framework matches the Agile Manifesto, 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 further identified those LeSS practices that could be used to mitigate the QRs challenges reported in (Alsaqaf et al., 2019).

Our results show that LeSS practices could be used to mitigate one or more QRs challenges. However, our study shows also that LeSS does not provide specific practices that could be used to mitigate some QRs challenges (e.g. *End user acceptance of QRs* and *Conceptual definition of QRs*).

This research has some practical implications. First, practitioners conscious about QRs in projects that employ LeSS, should make explicit steps towards creating practices that help counter issues due to unclear conceptual definitions of QRs (see Table 6). Also, practitioners should come up with their own ideas on how to manage the length of the QRs acceptance checklist, just because LeSS offers no specific help regarding this. On the other side, practitioners can rely on LeSS regarding coping with QRs challenges related to team coordination and communication. As LeSS is designed to support team collaboration, it seems relatively straightforward to resolve QRs issues traceable to team coordination, 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 et al., 2019).

REFERENCES

Abheeshta, P., Paasivaara, M., & Lassenius, C. (2018). *Benefits and Challenges of Adopting the Scaled Agile Framework (SAFe): Preliminary Results from a Multivocal Literature Review*. PROFES 2018, 9459, 334–351.

Agile Alliance. (2001). *Manifesto for Agile software development*, http://www.agilemanifesto.org.

Ambler, S. W., & Lines, M. (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), 1008–1017.

Alsaqaf, W., Daneva, M., & Wieringa, R. (2019). *Quality requirements challenges in the context of large-scale*

distributed agile: An empirical study. Information and Software Technology.

Beedle, M. (2018). *Enterprise Scrum Definition 4.0*. In Enterprise Scrum Inc.

Collab.net, & Versionone.com. (2019). *13th Annual State of Agile Report*. *VersionOne*. https://explore.versionone.com/state-of-agile.

Conboy, K., & Fitzgerald, B. (2004). *Toward a conceptual framework of agile methods: A Study of Agility in Different Disciplines*. XP/Agile Universe 2004, pp 105-116.

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

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*, 16(3), 287–298.

Helmy, W., Kamel, A., & Hegazy, O. (2012). *Requirements engineering methodology in agile environment*. International Journal of Computer Science Issues, 9(5 5-3), 293–300.

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

Krebs, J. (2008). Agile Portfolio Management. In Microsoft Press. 1st edition.

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*, Pearson Education, 1st edition.

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, H. (2017). *Scaling agile in organisaties*. In Van Haren Publishing.

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), 280–295.

Qumer, A., & Henderson-Sellers, B. (2006). *Measuring agility and adaptibility of agile methods: A 4-dimensional analytical tool*. IADIS International Conference on Applied Computing, January, 503–507. http://www.iadis.org.

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*. In Pearson, 1st edition.

Schwaber, K., & Sutherland, J. (2016). *The Scrum Guide*. In Scrum.Org.

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

Sutherland, J. (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.

West, D., Gilpin, M., Grant, T., & Anderson, A. (2011). *Water-scrum-fall is the reality of agile for most organizations today.* In Forrester Research.