Towards a Solution Proposal to Agile Quality Requirements Challenges in Large-scale Projects

Wasim Alsaqaf, Maya Daneva and Roel Wieringa

University of Twente, Semantics, Cybersecurity and Services Group, Enschede, The Netherlands

Abstract

Over the years, the growth in usage of agile approaches in large-scale and distributed projects revealed the range of possible challenges experienced by organizations on their agile transformation journey. This short paper is focused on one particular type of challenges, namely those concerning the quality requirements (QRs) in agile large-scale projects. Leveraging previously published empirical results on QRs challenges in this context, the present paper makes a proposal for a solution. Using Design Science research methodology, we propose the Agile Quality Requirements Elaboration (AQRE) approach which introduces (1) a new organizational role and (2) a two-step process to elaborate high-level goal(s) into epics and user stories alongside with QRs. The fitness and the usefulness of AQRE will be empirically evaluated as part of our future work.

Keywords

Agile, Quality requirements, Non-functional requirements, Large-scale distributed projects, Requirements engineering, Goal-oriented modelling, Design science, Focus group study

1. Introduction

In the last decades, the adoption of agile software development and project management approaches has grown rapidly in large-scale and distributed project contexts [1]. More than 20 scaling frameworks have been proposed and used to guide organizations in such contexts [2]. Several scaled frameworks, e.g. SAFe, LeSS and Disciplined Agile Delivery (DAD) have established themselves as "leaders" in the marketplace [8]. Despite the maturity of the existing scaled frameworks, their application does not always go smoothly [7], [8]. In particular, many studies have reported persistent challenges concerning the engineering of quality requirements (QRs) in agile context [9], [10], [11]. In our recent empirical work [10], we have identified 15 challenges that large-scaled and distributed agile (LSDA) projects cope with when it comes to QRs (see Table 1). Given this background, in the present short paper we set out to outline our proposed approach to engineer requirements in LSDA so that these 15 identified QRs challenges [10] could be mitigated. To come up with our proposal, we considered a research process grounded on Design Science [12] and drawing on concepts from goal-oriented requirements engineering (GORE). In what follows, we first summarize these concepts and then we introduce our solution proposal and the plan for its empirical evaluation. We note that we would not discuss the application of the Design Science methodology [12] as this is a short position paper and is bound to a page limit of 4 pages.

Agil-ISE'23: Intl. Workshop on Agile Methods for Information Systems Engineering, June 12, 2023, Zaragoza, Spain
EMAIL: w.h.a.alsaqaf@utwente.nl (W. Alsaqaf); m.daneva@utwente.nl (M. Daneva); r.j.wieringa@utwente.nl (R. Wieringa)
ORCID: 0000-0002-0253-428X (W. Alsaqaf); 0000-0001-7359-8013 (M. Daneva)


Table 1
The QR challenges as reported in [10]

Category	Challenges
1.Teams coordination and communication challenges	Late detection of QRs infeasibility
	Hidden assumptions in inter-team collaboration
	Uneven teams maturity
	Suboptimal inter-team organization
2. Quality assurance challenges	Inadequate QRs test specification
	Lack of cost-effective real integration test
	Lengthy QRs acceptance checklist
	Sporadic adherence to quality guidelines
3.QRs elicitation challenges	Overlooking sources of QRs
	Lack of QRs visibility
	Ambiguous QRs communication process
4.Conceptual challenges of QRs	Unclear conceptual definition of QRs
	Confusion about QRs specification approaches
5. Architecture challenges	Unmanaged architecture changes
	Misunderstanding the architecture drivers

2. Foundation for our Solution Proposal Design

Practitioners working in LSDA context (e.g. [5]) observed that in this context an agile project is usually part of a large IT initiative and rarely an isolated project. This observation has also been shared by the first author of this paper who has professional experience as an agile SCRUM master and software developer. IT initiatives are relatively stable and have predefined goals before one or more projects are launched within each initiative. Therefore, goal-oriented requirements engineering (GORE) as an analytical technique in the discipline of Requirements Engineering (RE) can be employed to analyze and decompose the (sub) goals of the IT initiatives. In the next subsections, we explain how LSDA and GORE fit together and what role the notion of goal modelling could possibly play in order to cope with QRs in LSDA projects.

2.1. Goal-Oriented Requirements Engineering (GORE)

Requirements engineering (RE) is the process of elaborating stakeholders' intentions into specifications of the desired system or services [14]. Therefore RE needs to understand those intentions that have to be fulfilled by the desired system of services. In RE, those stakeholders' intentions are referred to as goals [15]. Using goals to drive requirements has been popular [16] due to, among others, the following benefits [15]: (i) clarifying the context and the value of the system, (ii) driving and guiding the identification of the system requirements, since for each goal a set of requirements can be defined, (iii) giving the possibility for identifying and evaluating solution alternatives, (iv) giving guidelines to identify irrelevant requirements, (v) giving guidelines for requirements completeness, (vi) giving rational for the relevance of requirements, (vii) giving guidelines for identifying and resolving requirements' conflicts and (viii) giving stability since goals are not subject for frequent changes while requirements are. In alignment with Pohl [15], Daneva et al. [17] highlighted the importance of assessing stakeholders' goals early in the system development phase to achieve a clear scope definition which would guide the identification of the most significant requirements. Whether these benefits have been observed in LSDA projects and to what extent GORE adds value in that context has not been empirically researched in much depth. However, the analysis and refinement of a large-scale agile initiative might be a suitable application domain for goal-oriented RE methods for the following reasons. First, unlike user stories, initiatives have a longer time span and, as such, investing on the creation of and discussion around a

goal model may be rewarding. Second, one might think of the potential usefulness of GORE in LSDA, from the following perspective: in our previous work on QRs challenges in LSDA projects [10], we reported several mechanisms behind these challenges: (i) suboptimal priorities assigned to conflicted QRs, (ii) focusing too much on system's parts and losing the big picture, (iii) the emerging of relevant QRs late in the development phase. While these mechanisms were observed in LSDA projects, we acknowledge that they might not be unique to agile. As GORE was introduced to cope with such situations in 'traditional contexts', we thought that there would be no reason to assume that GORE would not work for agile projects. In fact, we believe that implementing GORE concepts could eliminate several of the mechanisms reported in our previous work [10], which in turn means that it can serve to mitigate the challenges in Table 1.

In GORE, goals can be of different levels of abstraction; for example, Cockburn [18] reported the following noticeable levels: 1) Cloud level – a high-level business goal that need to be decomposed into sub-goals, 2) Kite level – a decomposed goal from the Cloud level that represents an end-to-end system process, 3) Sea level – a user goal decomposed from the Kite level that can be achieved by one person within the end-to-end system process, and 4) Fish level – a task (not a goal by itself) carried out along with other tasks to achieve a user goal. Agile software development (ASD) however does not use these abstraction levels of goals. ASD uses actually the terms Themes, Initiatives, Epics and User Stories. Noreika et al. [19] defined those agile terms as follows: (i) Theme is a logical organization and aggregation of related user stories to show they have something in common and is managed by business representatives in a project. No software development activities are required to achieve themes. (ii) Initiative is a composition of epics that drive toward a common business goal which should not span more than one year. Initiatives provide also the needed context to help companies make decisions regarding the course of direction. Software development activities are required to achieve an initiative. (iii) Epic is a set of related user stories that need no longer than a quarter to be completed. To achieve Epics, software development activities are needed as well. (iv) User Story is the lowest level of granularity, that means work to be completed within one to four weeks. Similarly to Epics and Initiatives, user stories need software development activities to be implemented. Based on the aforementioned description, throughout this paper, we treat agile initiatives as high level business goals (e.g. Cloud), Epics − as sub-goals (e.g. Kite) and User stories − as user goals (e.g. Sea).

2.2. Modeling Requirements

Models have been used widely in software development. They provide an abstract representation of a particular complex problem to simplify the process of understanding the problem [20], [21]. More in detail, Muller et al. [20] and Pastor et al. [21] explained modeling as a simplification of a system to be built with an intended goal in mind and an abstraction of a relevant part while ignoring irrelevant aspects. Moreover, Girvan et al. [22] reported the following benefits of using models, namely: models provide (i) an effective manner for discussion and collaboration, and (ii) an effective medium for communications. These benefits are in line with the agile way of working where individuals' interaction and customer collaboration are highly valued [4]. The RE literature (e.g. [16]) introduced many GORE frameworks where goals are used to identify significant requirements and are, in turn, modelled. Specifically, the i* framework [23] has been recognized as one of the most popular to model goals [16]. Despite its broad use, the i* framework was experienced as not so easy to learn which hampered its adoption outside its community [24]. This experience forced the GORE community to come up with a response, which was in the form of the iStar 2.0 goal-based requirements modelling language [24]. Throughout this paper, we will use iStar to refer the iStar 2.0 as described in [24]. iStar is designed to provide means to model (i) different types of actors and their boundaries, (ii) independent elements e.g. goals, qualities, tasks and resources, and (iii) different types of relationships [24]. Further, we use iStar throughout this paper as a modelling tool to explain our proposed method.

3. Our proposal: the AQRE approach

The overall objective of our proposed method, called Agile Quality Requirements Elaboration (AQRE) approach, is to help agile teams deal with the QRs challenges reported in [10]. For our method

to work, it needs to be embedded into the larger software development process of an organization [25]. In our research context, this is the process of delivering a software system in a LSDA project. In line with this, we start on the premise that our proposed AQRE method should be executed before the start of the first sprint to decompose the high-level goal and help creating the product backlog for the project and then enable the start of this first sprint. The method can be used as well in future sprints to decompose the sub-goals of the high-level goal into smaller sub-goals. In a nutshell, our proposal, AQRE, represents a workshop session where participants discuss and elaborate a software development initiative into epics and then epics into user stories by using a goal-based requirements modelling language such as iStar. The AQRE approach consists of one mandatory role and two steps which refer to preparing and executing a QR-focused workshop. The role and workshop steps are described in the next subsections.

3.1. The AQRE role

Drawing on the industrial practice of McKinsey [26] and other large companies [27], AQRE introduces the organizational concept of Initiative Owner (IO). While this term has been used in industrial experience reports [26], [27], to the best of our knowledge the term IO has not been elaborated in sufficient depth in scientific studies. For example, although Bucy et al. [26] refer to this term, these authors did not provide a clear description of what they mean with it. Besides, Bucy et al. use the term IO in the context of organization's transition in general, and not related to agile in specific. Furthermore, Sutherland et al. [27] describe the role of IO as synonym to the role of Product Owner (PO). To avoid any confusion, in AQRE we use the term IO to refer to the one who is responsible for achieving the initiative (e.g. the high-level goal or goals) of the organization and coordinating the activities needed to implement the initiative in question. This initiative will be decomposed into Epic(s) and User Stories and assigned to one or more agile teams to be implemented. In case of multiple agile teams, each would have their own PO. Each PO is then responsible for coordinating the work of his/her own team based on the customer's values his/her team wants or deliver.

3.2. The AQRE Steps

Our proposed method includes (1) preparation of the AQRE workshop and (2) its execution. Below, we describe each of them in terms of activities that the AQRE participants would go through.

1) Preparation. The IO begins the initiative by providing a short description. The IO determines further who will be invited to the AQRE workshop (e.g. step two below) to discuss and elaborate the initiative. We note that the workshop participants should be chosen based on needed knowledge and not based on their role in the organization. The types of needed knowledge are: (i) domain knowledge, (ii) QRs knowledge, (iii) enterprise architecture knowledge, (iv) infrastructure and maintenance knowledge. Based on the nature of the initiative, other particular types of knowledge could be needed (e.g. security, regulations and compliance, usability). In that case, the IO invites the people with that particular knowledge as well. Besides, the IO also ensures that the workshop's participants have sufficient knowledge of goals decomposition techniques. If a participant has no prior exposure, then the IO organizes a training session as part of the preparation step to educate the participants on goal decomposition.

2) Execution of the Workshop. The IO starts the workshop by giving background information such as organization's vision and goals, the initiative's description and how it fits within the organization's vision and goals. Hereafter, the participants start decomposing the initiative using the AND/OR goals decomposition technique described in [15]. Prior to the workshop, the IO assured that the participants understand the goals decomposition technique (see step one on the previous page). The objective of this process is to break down the initiative into smaller goals (e.g. matching the possible epics and user stories) based on the expected value of the customer (e.g. the who), defining possible alternative (sub)goals or solution's directions, defining QRs associated with the identified (sub)goals, identifying and resolving conflicts between (sub)goals in general, and QRs in specific, and defining and agreeing upon the scope of the initiative. The workshop could take hour(s) or day(s), depending on how complex

the initiative is. Since our study focuses on the mitigation of the reported QRs challenges in [10], in the next sub-section we elaborate further on that subject.

3.2.1. QRs elaboration

During the process of breaking down (sub)goals, the participants have to identify those QRs associated with the identified (sub)goals. The identified QRs can be further broken down into other related QRs. For example, a security quality attribute could be decomposed into e.g. Confidentiality, Integrity and Authenticity. Performance, for example, could be broken down into e.g. Capacity and Resource Utilization. To guide this process optimally we advise the participants to use a QRs framework of their choice, e.g. the ISO25010 quality standards [13], the NFR framework [28], or the Sustainable Catalog [6]. After identifying and elaborating the QRs, the participants have to provide those QRs with enough details in order to enable the development teams to make the right design decisions (concerning these QRs) at the right time. This can be considered as "just-in-time requirements specification". The details should at least include: (i) the estimated impact of (fully) having or (partially) missing the QR on the related (sub)goal, and (ii) broad specification of the QRs. This means that the participants should specify the QRs in a way that does not prevent the development teams from being creative in making the right consideration towards the right implementation decision and in the same time gives the development teams enough boundaries to be able to implement and test the right QRs correctly [3]. For example, if the performance of collecting user data after logging in the system is an important QR, we can then specify what is an absolutely unacceptable performance like "user data should be collected and shown on screen in absolutely no longer than 7 seconds". Instead of "user data should be collected and shown on screen in 5 seconds". The last option will limit the decisions which the development teams can make, while the first option will give the development teams space to make their consideration, especially if there are conflicting performance and usability requirements, e.g. the a mount of data that should be shown after logging in the system.

4. Conclusion and further work

In this paper we proposed the AQRE method for elaborating requirements in general and QRs in specific from high-level goal(s) in LSDA projects. The proposed method consists of one role (e.g. IO) and two steps (e.g. Workshops preparation, Workshop execution). Adhering to the agile principles is also taken into consideration since the method leans on collaborations and individuals interactions. The primary objective of AQRE is to identify a remedy to the QRs challenges reported in [10]. We plan to empirically evaluate the fitness and usefulness of the proposed method as part of our further work. This includes a series of structured focus groups with practitioners working in LSDA context. Specifically, we plan three focus groups. One is in a large government organization in the Netherlands. The second is in a consulting company where agile consultants have exposure to a variety of business sectors and organizations adopting agile scaling frameworks in LSDA contexts. The third is with practitioners from a national professional organization of agile requirements engineers. Our aim form conducting these focus groups is to understand to which extend our proposed method (e.g. AQRE) could mitigate the QRs challenges reported in [10].

5. References

[1] Digital.ai Software, "15th State of Agile Report," *Digital.ai*, pp. 1–23, 2021.

[2] Ö. Uludağ, P. Philipp, A. Putta, M. Paasivaara, C. Lassenius and F. Matthes, "Revealing the state of the art of large-scale agile development research: A systematic mapping study," *Journal of Systems and Software*, vol. 194, no. 3, pp. 212–220, 2022.

[3] S. Lauesen, Software Requirements: Style and Techniques, First Edit. Pearson Education, 2002.

[4] Agile Alliance., Manifesto for Agile software development. 2001.

[5] S. W. Ambler and M. Lines, Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise. IBM Press, 2012..

[6] D. Albuquerque, A. Moreira, J. Araujo, C. Gralha, M. Goul, and I. S. Brito, "A Sustainability Requirements Catalog for the Social and Technical Dimensions," in ER, 2021, vol. 1, pp. 381–394.
[7] J. Smart, "To Transform to Have Agility, Dont Do a Capital A, Capital T Agile Transformation," *IEEE Softw*, vol. 35, no. 6, pp. 56–60, 2018.
[8] K. Conboy and N. Carroll, "Implementing Large-Scale Agile Frameworks: Challenges and Recommendations," *IEEE Softw*, vol. 36, no. March/April, pp. 1–9, 2019.
[9] R. R. Maiti and F. J. Mitropoulos, "Capturing , Eliciting , Predicting and Prioritizing (CEPP) Non-Functional Requirements Metadata During the Early Stages of Agile Software Development," in *SoutheastCon*, 2015.
[10] W. Alsaqaf, M. Daneva, and R. Wieringa, "Quality requirements challenges in the context of large-scale distributed agile: An empirical study," *Inf Softw Technol*, Feb. 2019.
[11] I. Inayat, S. Salwah, S. Marczak, M. Daneva, and S. Shamshirband, "A systematic literature review on agile requirements engineering practices and challenges," *Comput Human Behav*, vol. 51, no. October, pp. 915–929, 2014.
[12] R. J. Wieringa, Design Science Methodology for Information Systems and Software Engineering, Springer Heidelberg, 2014.
[13] ISO/IEC, "ISO/IEC 25010:2011 - Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models," 2011, [Online]. Available: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=35733
[14] A. Van Lamsweerde, R. Darimont, and E. Letier, "Managing Conflicts in Goal-Driven Requirements Engineering," vol. 24, no. 11, pp. 908–926, 1998.
[15] K. Pohl, Requirements Engineering Fundamentals, Principles, and Techniques, 1st ed. Springer Berlin, Heidelberg, 2010.
[16] J. Horkoff, F.B. Aydemir, E. Cardoso, T. Li, A. Maté, E. Paja, M. Salnitri, L. Piras, J. Mylopoulos and P. Giorgini, "Goal-oriented requirements engineering : an extended systematic mapping study," in *RE*, 2019, pp. 133–160.
[17] M. Daneva, M. Kassab, M. L. Ponisio, R. J. Wieringa, and O. Ormandjieva, "Exploiting a Goal-Decomposition Technique to Prioritize Non-functional Requirements," in *10th WER*, 2007.
[18] A. Cockburn, *Writing Effective Use Cases*, First. Addison Wesley, 2000.
[19] K. Noreika and S. Gudas, "Modelling the alignment between agile application development and business strategies," in *BIR-WS*, 2021, vol. 2991, pp. 59–73.
[20] P. Muller, F. Fondement, B. Baudry, and B. Combemale, "Modeling modeling modeling," *Softw Syst Model*, vol. 11, no. 3, pp. 347–359, 2012.
[21] O. Pastor, A. Pierantonio, and G. Rossi, "Teaching Modeling in the time of Agile Development," *Computer (Long Beach Calif)*, vol. 55, no. June, pp. 73–76, 2022.
[22] L. Girvan and D. Paul, *Agile and Business Analysis Practical guidance for IT professionals*, First. Bcs Learning & Development Limited, 2017.
[23] E. S. K. Yu, "Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering," in *ISRE*, 1997, pp. 226–235.
[24] F. Dalpiaz, X. Franch, and J. Horkoff, "iStar 2.0 Language Guide." pp. 1–15, 2016.
[25] S. Pfleeger and J. Atlee, *Software Engineering: Theory and Practice*, 4th ed. Pearson, 2009.
[26] M. Bucy, T. Fagan, B. Maraite, and C. Piaia, "Keeping transformations on target," *McKinsey & Company*, no. March, pp. 1–17, 2017. [Online]. Available: https://www.mckinsey.com/capabilities/rts/our-insights/keeping-transformations-on-target
[27] J. Sutherland, D. Rigby, and H. Takeuchi, "Embracing Agile: How to master the process that's transforming management," *Harvard Business Review*, vol. 94, no. 5, pp. 40–50
[28] L. Chung, B. A. Nixon, E. YU, and J. Mylopoulos, *Non-Functional Requirements in Software Engineering*, Springer NY, 2000.