Agile Quality Requirements Management Best Practices Portfolio: A Situational Method Engineering Approach

Lidia López[1(✉)], Woubshet Behutiye[2], Pertti Karhapää[2], Jolita Ralyté[3], Xavier Franch[1], and Markku Oivo[2]

[1] Universitat Politècnica de Catalunya (UPC), Barcelona, Spain
{llopez,franch}@essi.upc.edu
[2] University of Oulu, Oulu, Finland
{woubshet.behutiye,pertti.karhapaa,markku.oivo}@oulu.fi
[3] University of Geneva, Geneva, Switzerland
jolita.ralyte@unige.ch

Abstract. Management of Quality Requirements (QRs) is determinant for the success of software projects. However, this management is currently under-considered in software projects and in particular, in agile methods. Although agile processes are focused on the functional aspects of the software, some agile practices can be beneficial for the management of QRs. For example, the collaboration and interaction of people can help in the QR elicitation by reducing vagueness of requirements through communication. In this paper, we present the initial findings of our research investigating what industrial practices, from the agile methods, can be used for better management of QRs in agile software development. We use Situational Method Engineering to identify, complement and classify a portfolio of best practices for QR management in agile environments. In this regard, we present the methodological approach that we are applying for the definition of these guidelines and the requirements that will lead us to compile a portfolio of agile QR management best practices. The proposed requirements correspond to the whole software life cycle starting in the elicitation and finalizing in the deployment phases.

Keywords: Quality requirement · Non-functional requirement · Agile development · Situational Method Engineering

1 Introduction

Agile methods are becoming increasingly popular in the software industry [1–3]. Customer satisfaction through early and continuous delivery of valuable software, adaptability to late requirements changes, short and iterative development cycles are some principles of agile software development (ASD) methods [4]. Another important aspect of software development that has attracted a lot of attention is software quality, mainly represented by the quality requirements (QRs; also referred to as non-functional requirements –NFRs) of the product [5]. However, it has been documented that the management of QRs in software development in general [5] and in ASD in particular [6] is problematic, e.g. important QRs might be neglected in ASD [7].

M. Felderer et al. (Eds.): PROFES 2017, LNCS 10611, pp. 548–555, 2017.
https://doi.org/10.1007/978-3-319-69926-4_45

One aspect of ASD is that agile principles put emphasis on communication and linking of people [4]. The closer collaboration between people within a development team, e.g. requirements engineers and testers, helps in generating an understanding of the requirements so that development can progress and testing can be conducted properly despite lower quality of the requirements and lack of documentation [8]. Agile practices can also help the QR elicitation by reducing vagueness of requirements through communication [9], QRs in particular, since defining good, verifiable, and complete QRs is quite difficult.

Improving the management of QRs in agile projects is the ultimate goal of the Q-Rapids (Quality-aware Rapid Software Development) project[1]. In order to achieve this goal, we aim at defining a set of guidelines for integrating QR management into the ASD life cycle. There are several methods, techniques and models that can be applied for managing QRs, making difficult the definition of a unique method to be applied in any organization. In the context of ASD, Qumer and Henderson-Sellers applied Situational Method Engineering (SME) to create a software development method combining agile and formal practices in a large software development organization [10]. Following the same approach, in this paper we propose using SME to identify, complement, and classify a portfolio of best industrial practices in order to define a method for QR management in agile environments.

The rest of the paper is organized as follows. Section 2 introduces the research approach followed, including the background necessary to apply SME. The construction of the method is based in the software development process detailed in Sect. 3. Section 4 includes the definition of the method requirements, and Sect. 5 includes an example of the guidelines associated to the QRs prioritization. Finally, Sect. 6 concludes the presentation of the work included in this paper and discusses our future work.

2 Situational Method Engineering

2.1 Background

In this work we apply the assembly-based Situational Method Engineering (SME) approach [11] as underpinning theory for capitalizing best practices in the domain of QR management in ASD, and for reusing them in the construction of situation-specific methods. Following this approach, the knowledge of such methods has to be formalized in terms of reusable method chunks. A method chunk describes the method process (i.e., the guidelines) and its related products (i.e., the concepts and artefacts used/transformed/created by applying the guidelines). It also specifies the situation in which it can be applied (i.e., the required input artefacts) and the intention (i.e., the engineering goal) to be reached. The method chunks are used as building blocks for constructing a situation-specific method, which can be a project-specific method or even a configurable method family including several method chunk variants for each method step. In both cases, the approach consists of defining method requirements and then selecting and assembling method chunks satisfying these requirements. Method requirements (also

[1] http://q-rapids.eu/.

called requirements map) are specified as a desired process model by using the Map process modeling formalism [12], which allows to express methods in terms of intentions and strategies to reach the intentions. The variability and flexibility of a method is reached by defining several strategies for achieving an intention.

The sources for engineering method chunks can be various: existing methods, standards, templates, and best practices. Depending on their formalization and level of detail, the creation of method chunks consist in reengineering the existing method knowledge or defining it from scratch.

2.2 Application

The assembly-based SME approach has been applied in various software and information systems engineering domains. For instance, Ralyté et al. reengineered the RESCUE Requirements Process into a modular method (a collection of method chunks organized into a multi-level process map) allowing to assess the quality of the method, to identify omissions and weaknesses, and to reason about its improvements [13]. This case also demonstrated the effectiveness of the SME approach for modelling large-scale engineering processes. In a different domain, López et al. presented the OSSAP method [14], applying assembly-based SME approach to construct a method for OSS adoption business processes. The OSSAP chunks correspond to the different ways of adopting OSS and the pieces of processes to be adopted by the organization, depending on the way they want to be involved with the OSS community producing the OSS.

3 Software Development Process in Agile Projects

In this section, we present the analysis of the software development process employed in four use cases (UCs) of the Q-Rapids project. The results are based on preliminary findings of case studies conducted to understand the software development processes and QR management practices adopted in selected projects of the Q-Rapids industrial partners. The Q-Rapids industrial partners are representatives of small, medium, and large sized companies from three different countries (Finland, France and Poland), all produce software in different domains (telecommunications, secure solutions, modeling and ad-hoc solutions). Qualitative analysis was done on the 12 semi-structured interviews conducted in the UCs to get an understanding of the development processes.

Our findings reveal that all of the UCs adopt variants of Scrum tailored to their specific context of development. The UCs operate in predefined release cycles that range from two weeks to six months. The sprint cycle varied from one to four weeks. Medium and large companies are characterized by complex backlog structure and multiple teams. The smaller companies utilized a single backlog and consist of a small sized team. Additionally, the ASD maturity level applied in the UCs also varied. We observed both similarities and differences in the practices, roles and tools utilized in the UCs.

During initial stages of the development process, the UCs elicit requirements (both functional requirements and QRs) mainly based on customer needs. At this stage, high level features are elicited together with the customer. Features that bring more value to

the customer are prioritized. However, the level of customer involvement, as well as the practices and roles involved in the process, varies among the UCs. For instance, two UCs from small and medium sized companies mainly utilize the customer for eliciting requirements. The other two UCs from medium and large sized companies consider additional factors such as product roadmaps, the status of the market and problems of potential customer segments. Roles involved in higher level requirements elicitation included product owners, product and technical managers, sales team, and usability experts. Product and technical managers made requirements prioritization decision in UCs of medium and large companies. On the other hand, smaller companies relied on the product owners' decisions for requirements elicitation and prioritization. Elicitation of the higher-level features considered both functional requirements and QRs.

The higher level features are refined and specified into lower level features or user stories and tasks. In medium and large organizations, higher level features were refined in several steps due to the product size. On the other hand, in smaller companies, the number of refinement steps were fewer.

Communication happens throughout the development process in all of the UCs. Face-to-face communication serves as the main source of communication in small sized companies. In such cases, face-to-face communication facilitates the development process, as the developers are close to each other and usually in the same room. Additionally, there was less emphasis on the documentation practices. However, in medium and large sized companies, documentation and shared tools serve as sources of communication. Face-to-face communication was adopted only at lower (local) level.

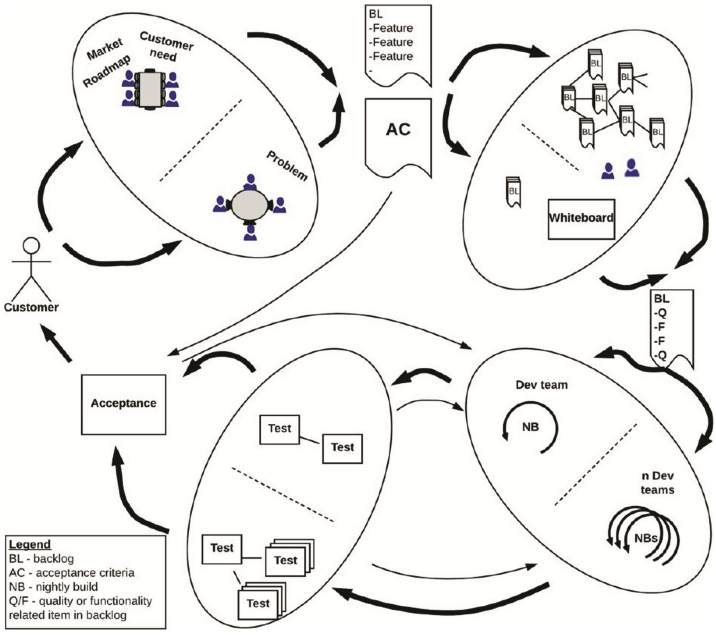

Fig. 1. Aggregated view of the development processes in the UCs

All UCs employ continuous integration in their development process. Nightly builds, integration tests, and acceptance tests are applied in the verification and validation process. The testing practices also varied with the size of the companies. Figure 1 depicts the generic view of the development process adopted in the UCs.

4 QR Management Method Requirements

The analysis of the software development process of the UCs, described in the previous section, uncovered that they do not use a predefined existing method for QR management. The organizations use and combine different methods and techniques in different ways for setting their own agile oriented development process. The aim of this work is setting up a portfolio of best practices organizing and complementing these techniques to improve QR management in the context of ASD processes.

Due to this diversity of methods and techniques, we are developing this portfolio applying SME, concretely creating a new method constructed from scratch [15]. In order to identify the needed guidelines, we applied a process-driven strategy to elicit the method requirements, which is more relevant in the case of a new method construction [16]. In order to specify the requirements for the method, we need to (1) identify the set of intentions related to the QR management in the current processes, and (2) identify the possible strategies for fulfilling these intentions.

During the UCs analysis, we collected the initial set of intentions to be fulfilled by the new method: *Elicit*, *Specify*, *Communicate,* and *Verify and Validate* QRs. These intentions correspond to the underlying goals for each activity of the generic development process depicted in Fig. 1: meetings discussing market roadmap and customer needs for *elicitation*, backlogs and whiteboards for *specification* and *communication*, and testing for *verification and validation*. Then, we complemented the set of intentions identifying the different strategies to fulfill them. The intentions are represented as nodes and strategies as edges in the requirements map shown in Fig. 2.

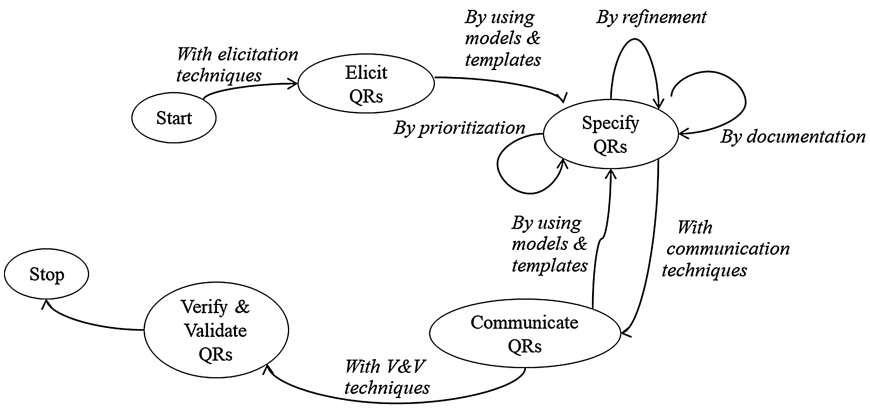

Fig. 2. QR management method requirements map

Most of the strategies included in the requirements map are still generic, except for the strategies to fulfill the *Specify QR* intention. The Q-Rapids UC providers (see Sect. 3), pointed out that we can find different levels of requirements in ASD processes, from high-level requirements (coming from the elicitation activity) to lower-level requirements (defined in later stages), which are the refined requirements that can be translated to user stories, features or tasks to be communicated to the development team. Therefore, refinement is the strategy to specify new lower-level requirements. Prioritization is really important in agile environments, requirements need to be arranged by priority to be fully specified before they are communicated to the development teams.

5 Example: Chunks for QR Prioritization

In this section, we describe the possible strategies for fulfilling the *Prioritize QRs* intention. From the analysis of the UC processes, we identified the following two situations: the prioritization by urgency (issue-driven) and prioritization based on value (value-driven). The prioritization by urgency occurs when some blocking situation arises during the software development process that affects the expected workflow. For example, if there is a specific problem/issue in the development of a critical feature, the development team should reprioritize the work focusing on fixing this situation. On the other hand, when no critical situations should be handled, the organization can prioritize their requirements with no specific problem to solve.

For the value-driven strategy, we identified an existing method chunk included in [17] for cost-value requirements prioritization. This value-driven prioritization chunk proposes having two criteria for evaluating requirements: relative value and relative cost, which are used for ranking the requirements. Figure 3 reproduces the process map for this chunk.

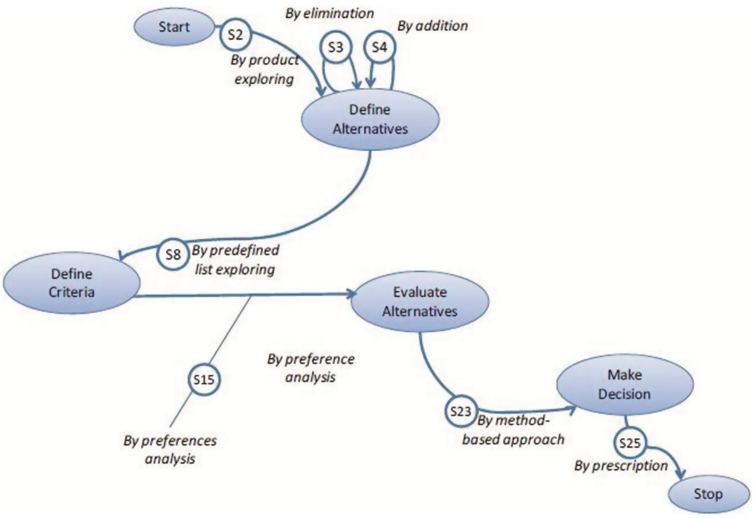

Fig. 3. Cost-value requirements prioritization approach chunk [17]

We did not find any existing method for the Issue-driven prioritization, so we envisage that we are going to create one. It could be based on the idea of identifying the features related to the issue, and then the dependencies for this feature, the features would be ranked depending on the dependency to the critical issue to solve.

According to SME process, we refined the strategy named "by prioritization" into two: Value-driven prioritization and Issue-driven prioritization.

6 Conclusions and Future Work

Organizations do not use a predefined existing method for QR management. In this paper we present the initial findings of our research investigating what industrial practices, from the agile methods, can be used for better management of QRs in agile software development.

In this paper, we present how we are using Situational Method Engineering (SME) to identify, complement and classify a portfolio of best practices for agile QR management. SME is used to construct methods that can be customized to fulfill the organization needs. The first results reported in this paper correspond to the initial set of intentions that are leading our method requirements elicitation. The guidelines should include best practices to fulfill four different intentions: *QR elicitation*, *specification*, *communication,* and *verification and validation*, and the three strategies for fulfilling the specification intention: *by refinement, documentation* and *prioritization*. So far, we identified two concrete strategies for the prioritization: the prioritization by urgency (issue-driven) and prioritization based on value (value-driven), and the paper includes the method chunk corresponding to the value-driven strategy.

We are in the initial stages of identification of different strategies to achieve identified intentions. Our future work is to select current strategies and create new ones to produce a complete set of chunks that will shape our best practices portfolio.

Acknowledgments. This work is a result of the Q-Rapids project, which has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement N° 732253.

References

1. Cristal, M., Wildt, D., Prikladnicki, R.: Usage of Scrum practices within a global company. In: IEEE International Conference on Global Software Engineering, ICGSE 2008, pp. 222–226
2. Hamed, A.M.M., Abushama, H.: Popular agile approaches in software development: review and analysis. In: 2013 International Conference on Computing, Electrical and Electronics Engineering (ICCEEE), pp. 160–166 (2013)
3. Matharu, G.S., Mishra, A., Singh, H., Upadhyay, P.: Empirical study of agile software development methodologies: a comparative analysis. ACM SIGSOFT Softw. Eng. Notes **40**(1), 1–6 (2015)
4. Fowler, M., Highsmith, J.: The agile manifesto. Softw. Dev. **9**(8), 28–35 (2001)

5. Nuseibeh, B., Easterbrook, S.: Requirements engineering: a roadmap. In: Proceedings of the Conference on the Future of Software Engineering, pp. 35–46 (2000)
6. Schön, E.M., Thomaschewski, J., Escalona, M.J.: Agile requirements engineering: a systematic literature review. Comput. Stand. Interfaces **49**, 79–91 (2017)
7. Ramesh, B., Cao, L., Baskerville, R.: Agile requirements engineering practices and challenges: an empirical study. Inform. Syst. J. **20**(5), 449–480 (2010)
8. Uusitalo, E.J., Komssi, M., Kauppinen, M., Davis, A.M.: Linking requirements and testing in practice. In: 16th IEEE International Requirements Engineering, RE 2008, pp. 265–270 (2008)
9. Inayat, I., Salim, S.S., Marczak, S., Daneva, M., Shamshirband, S.: A systematic literature review on agile requirements engineering practices and challenges. Comput. Hum. Behav. **51**, 915–929 (2015)
10. Qumer, A., Henderson-Sellers, B.: Construction of an agile software product-enhancement process by using an Agile Software Solution Framework (ASSF) and situational method engineering. In: Annual International Computer Software and Applications Conference (COMPSAC), pp. 539–542 (2007)
11. Ralyté, J., Rolland, C.: An assembly process model for method engineering. In: Dittrich, K.R., Geppert, A., Norrie, M.C. (eds.) CAiSE 2001. LNCS, vol. 2068, pp. 267–283. Springer, Heidelberg (2001). doi:10.1007/3-540-45341-5_18
12. Rolland, C., Prakash, N., Benjamen, A.: A multi-model view of process modelling. Requir. Eng. J. **4**(4), 169–187 (1999)
13. Ralyté, J., Maiden, N., Rolland, C., Deneckère, R.: Applying modular method engineering to validate and extend the RESCUE requirements process. In: Delcambre, L., Kop, C., Mayr, H.C., Mylopoulos, J., Pastor, O. (eds.) ER 2005. LNCS, vol. 3716, pp. 209–224. Springer, Heidelberg (2005). doi:10.1007/11568322_14
14. López, L., Costal, D., Ralyté, J., Franch, X., Méndez, L., Annosi, M.C.: OSSAP – a situational method for defining open source software adoption processes. In: Nurcan, S., Soffer, P., Bajec, M., Eder, J. (eds.) CAiSE 2016. LNCS, vol. 9694, pp. 524–539. Springer, Cham (2016). doi: 10.1007/978-3-319-39696-5_32
15. Henderson-Sellers, B., Ralyté, J.: Situational method engineering: state-of-the-art review. J. Univ. Comput. Sci. **16**(3), 424–478 (2010)
16. Ralyté, J., Deneckère, R., Rolland, C.: Towards a generic model for situational method engineering. In: Eder, J., Missikoff, M. (eds.) CAiSE 2003. LNCS, vol. 2681, pp. 95–110. Springer, Heidelberg (2003). doi:10.1007/3-540-45017-3_9
17. Kornyshova, E., Deneckère, R., Rolland, C.: Method families concept: application to decision-making methods. In: Halpin, T., Nurcan, S., Krogstie, J., Soffer, P., Proper, E., Schmidt, R., Bider, I. (eds.) BPMDS/EMMSAD -2011. LNBIP, vol. 81, pp. 413–427. Springer, Heidelberg (2011). doi:10.1007/978-3-642-21759-3_30