


# Scaling up the Planning Game: Collaboration Challenges in Large-Scale Agile Product Development

Felix Evbota<sup>1,2</sup>, Eric Knauss<sup>1,2(</sup>), and Anna Sandberg<sup>3</sup>

<sup>1</sup> Department of Computer Science and Engineering, Chalmers University of Technology, Gothenburg, Sweden

<sup>2</sup> Department of Computer Science and Engineering, University of Gothenburg, Gothenburg, Sweden

fevbota@gmail.com, eric.knauss@cse.gu.se

<sup>3</sup> Ericsson AB, Gothenburg, Sweden

**Abstract.** One of the benefits of agile is close collaboration of customer and developer. This ensures good commitment and excellent knowledge flows of information about priorities and efforts. However, it is unclear if this benefit can be leveraged at scale. Clearly, it is infeasible to use practices such as planning game with several agile teams in the room. In this paper, we investigate how a large-scale agile organization manages, what challenges exist, and which opportunities can be leveraged. We found challenges in three areas: (i) the ability to estimate, prioritize, and plan; (ii) the context of planning with respect to working environment, team build-up, and team spirit; and (iii) the ceremonial agreement which promises to allow leveraging abilities in a given context.

**Keywords:** Large-scale agile · Planning · Collaboration · Communication

## 1 Introduction

One of the advantages associated with agile software development is the focus on customer collaboration and the ability to deliver customer value quickly and incrementally [8]. Popular agile methods such as Scrum [18] and eXtreme Programming (XP) [17] have powerful planning mechanisms in place, around practices such as backlog grooming, distinction between product and sprint backlog, and defining Sprint goals in Scrum, or user stories, onsite customer, acceptance testing, and planning game in XP. These practices facilitate excellent information flows: Agile development teams learn about priorities of customers, while customer representatives (product owner or onsite customer) gain knowledge about feasibility and costs of implementing their needs.

Consequently, agile methods have been applied to more and more complex development endeavors, including large and embedded software systems [1, 7]. In such contexts, it is necessary to scale up agile principles and even though this is not an easy task to do, successes have been reported, especially on reducing time-to-market of features or average times for solving customer requests [1].

Despite these successes, challenges remain, e.g. in coordination and communication of large teams [10]. In this paper we present a qualitative case study based on ten semi-structured interviews that explores how program and project leaders, product owners, line managers, and developers of cross-functional teams coordinate around planning work in a large-scale agile setting.

**Contribution.** Our contribution in this paper is two-fold. First, we provide insights about the challenges of aligning the views of product owners and product developers during planning in large-scale agile. Secondly, we provide a model on the relationship of different challenge types that shows how technical abilities (e.g. to estimate, to prioritize, or to plan) depend on contextual aspects (such as team build-up, work environment, and team spirit). Our study indicates that ceremony agreement plays a crucial role for enabling technical abilities of estimation, prioritization, and planning in a given context defined by the team structure and its environment.

## 2 Background and Related Work

Agile software development is incremental, cooperative, and adaptive [6] and facilitates responding to change quickly and frequently [8]. According to Leffingwell [1] it leads to the following business benefits: increase in productivity, increase in team morale and job satisfaction, faster time to market, and increase in quality. In this paper, we refer to the agile methods Scrum and XP [1, 8] and we are inspired by the XP practice Planning Game, which suggest that developers, managers, and customers meet at the start of each iteration to estimate and prioritize requirements (user stories) for the next release [6, 8].

Agile methods rely heavily on face-to-face communication [3, 5, 9]. However, if the number of the involved developers (and teams) grows, it becomes extremely difficult to practice face-to-face communication between different teams [3, 5, 9]. Such growth is usually triggered by a large number of complex requirements and there is a considerable challenge to manage them [3, 4]. While Larman and Vodde suggest the use of area product owners to scale the product owners role [2], Lassenius and Paavi report that collaboration and communication between teams and product owners in such a setup were challenging and did not work well [4]. These challenges are related to the fact that area product owners work with different teams and that teams could receive different user stories from several area product owners, which thereby become difficult to prioritize [4]. Also, the introduction of Scrum of Scrums meeting (SoS— a meeting where the Scrum masters of all Scrum teams meet on a daily or weekly basis to discuss the state of every team) was found to be ineffective because of the large number of Scrum masters that were involved. As a consequence of this large audience, it was difficult to get everybody interested in the coordination meetings [4].

Products with long lifecycle (e.g. complex systems like ships or planes) tend to have very comprehensive backlogs [20]. According to Larman and Vodde, it is the responsibility of the product owner to prioritize the product backlog to improve return on investment or delivery of value [20]. For this, they suggest using planning poker to assign effort and relative value points (RVP) as a lightweight proxy for ‘value’ (e.g. on a scale of 1–7). The product owner then prioritizes items based on low effort estimate and high

RVP as well as other factors, such as stakeholders' preferences, strategic alignment, impact on profit, and risk [20]. Daneva et al. describe requirements prioritization in large-scale agile software development as a decision-making process [19]. In this process, priority drives the packaging of requirements into releases: requirements with highest priorities are packaged for development first [13] and usually the main criterion for such prioritization is the business value of clients and vendors [19]. Agile development however implies incremental development prioritization, which is hard to maintain: Priorities of requirements change, so there is always a need to update the priority list [13]. Thus, for being able to carry out a successful decision in prioritization, it is of paramount importance to continuously consider the roles involved, the contextual setting of the prioritization process, the prioritization criteria being used, and the kind of trade-offs being made [19]. Yet, according to Daneva et al. [19], there is a lack of (empirical) knowledge about how requirements prioritization is done in large-scale agile software development [19], a gap that we aim at exploring in this paper.

### 3 Research Method

The setting for this case study is Ericsson AB in Gothenburg, Sweden. Ericsson was founded in the year 1876 and is a world leading Swedish telecommunication company. Ericsson has its headquarter in Stockholm in Sweden and has more than 110, 000 employees from different parts of the world such as Sweden, China, United States of America, South Africa, United Kingdom, Germany, Nigeria, and other countries. Ericsson is involved in the development of several products, such as, cable TV, network systems, Internet Protocol, networking equipment, video systems, mobile, and fixed broadband. Ericsson also renders extensive services to its customers. Ericsson uses agile methods at a large-scale in development of their products and their methods include for example Scrum, Extreme programming, and Kanban.

This paper presents a case study conducted at Ericsson in which we explore collaboration challenges during planning of large-scale agile development efforts. We choose a qualitative case study approach that allows studying large-scale agile planning in its context [12, 15, 16]. Accordingly, we selected ten participants based on their ability in order to cover the following roles: operative product owner (OPO), line manager, program leader, project leader, release leader, team leader and developer. Through this setup, we were able to investigate collaboration challenges between teams, product owners, and program leaders in large-scale agile software development in depth. Some of our participants have about 30 years of working experience at Ericsson. The interviews were based on an interview guide with open-ended and probing questions [15]. The interview questions focus on the collaboration challenges that teams, operative product owners, and program leaders face while planning, estimating, prioritizing, and delivery of features/tasks both on the teams and program levels at Ericsson.

We carried out a verbatim transcription of the interviews data that we collected and analyzed it qualitatively to form themes and identify patterns using the six steps suggested by Braun and Clarke's thematic analysis approach [11]:

*Step 1: Familiarizing with your data.* We transcribed the interviews and read it several times to familiarize ourselves with the data.

*Step 2: Generating initial codes.* We highlighted quotes in the transcriptions that related to our research and assigned initial codes.

*Step 3: Searching for themes.* We grouped all codes from phase two into a number of themes.

*Step 4: Reviewing the themes.* We reviewed the candidate themes from phase three several times and created a thematic map containing seven categories of challenges as well as some sub challenges.

*Step 5: Defining and naming themes.* We further reviewed the themes we generated from phase four by checking them with interview data and the codes generated from previous phases. Codes that we found assigned to wrong themes were moved to their rightful themes. In addition to that, we also reviewed the names we gave to the themes based on the sub challenges we have in each of the themes to ascertain suitable and distinctive naming.

*Step 6: Producing the report.* For this paper we further analyzed the themes to identify key challenges of large-scale agile planning in order to present and discuss our findings.

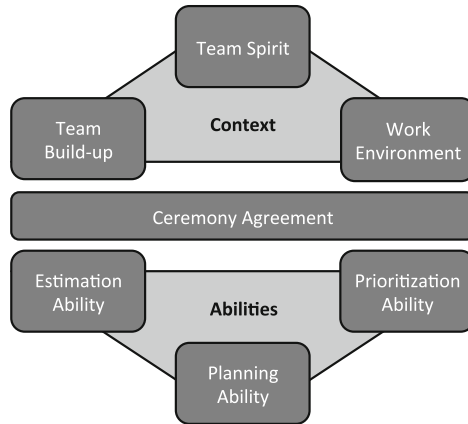
In our analysis we coded the interview data by taking our research question into consideration [11]. The analysis of the interview data was not linear, meaning that we did not always follow the suggested steps in their exact order. The analysis phase was instead recursive; meaning that while we were in Step 3 for example, we often had to revisit Step 2 and even Step 1.

## 4 Findings: Technical Abilities, Context, and Ceremonies

Our research method resulted in a number of observations that can be arranged into seven major themes, as shown in Fig. 1: The technical *ability to estimate, prioritize, and plan*, the context of planning in terms of *team build-up, work environment, and team spirit*, and finally the *ceremony agreement* that plays a key role in aligning technical abilities and context. In this section, we will describe our findings with respect to each of the themes before we will discuss relationships between the themes as well as their implications to research and practice in Sect. 5.

### 4.1 Technical Ability Challenges

For adequate planning, agile teams (regardless of their scale) need to bring together the *ability to estimate* required work, *prioritize* it with respect to business value, and to combine this knowledge into a good *plan* for the coming iteration(s). At a large-scale, where a hierarchy of product owners manages backlog items for a large number of teams, we identify (communication) challenges in all three parts.



**Fig. 1.** Relevant themes of large-scale agile planning concern technical abilities as well as context of planning. Ceremony agreement plays a key role to connect both spheres.

**Estimation Ability.** According to our interviewees, it is extremely challenging to make a long-term estimation (i.e. making estimates for several months) because of the amount of content (i.e. product backlog items) is too big. This experience has lead teams to become *skeptical about estimation* in general. In addition, the fast pace and large-scale leads to a significant amount of troubleshooting, which is hard to anticipate and impacts available resources during a sprint:

*“[Previously we] estimated on available days in the sprint, that is not a good way because you do not include the unexpected things” [OPO]*

Another challenge with estimation in large-scale agile is the *need to monitor discussions* during story estimation. Our interviewees reported that without systematic monitoring of such discussions, they could go on in circles for hours without making significant progress.

In the context of cross-functional teams, an unstructured estimation session can also lead to a pathological situation, where team members should *estimate tasks that do not fit their role*, as for example shown by the following quote from one of our interviewees:

*“[Sometimes we have a] tester estimating design tasks and a designer estimating test tasks. It is important to know whose estimation should be looked at”. [Line manager]*

Most of our interviewees stated that they are not experts in estimation and the challenges of large-scale agile estimation create a steep learning curve, especially for new team members. Estimation is based learning from past iterations and experience in the team.

**Prioritization Ability.** Large-scale agile product development implies a more or less *complex structure of product owners and backlogs* on different levels (such as total product, product area, operational level). It is hard to establish a shared vision with so many stakeholders, leading to disagreements and continued discussions about priorities.

*“The challenge is if you have a lot of small backlog you are not in control at all because if you have one common backlog and you decide on a program level, that is how we work [...] if not everything is visible on the common backlog program and only visible in the XFTs backlog then you maybe having a mismatch.” [OPO]*

Such potential *inconsistencies between different backlogs* as well as *lack of their transparency are a big challenge*. According to our interviewees, prioritization decisions need to be done on the program level. In contrast, information that is only available in the local backlog of a cross-functional team cannot be accessed and taken into consideration. Thus, if significant information is only available on the team’s level and not visible on the program level, prioritization decisions cannot be optimal and mismatches will be the consequence. It is impossible to share all information (e.g. about decisions, new technology, dependencies) with other teams, product owners, and the whole organization. It is also very hard to understand the significance of information on the team level for prioritization. This potential mismatch makes it hard for the program board to establish a prioritization that teams will follow.

**Planning Ability.** Under the theme planning ability we discuss findings that either affect both estimation and prioritization, or that arise from translating estimations and prioritizations into a concrete plan. The planning ability in large-scale agile product development is according to our interviewees enabling to balance market priorities with the inflow of change requests and bug reports. Our interviewees mentioned two important goals: To be (i) less release-focused in their planning, thus supporting continuous deployment to customers, and (ii) to achieve a higher flexibility in their planning. For both goals, the existing challenges are impediments. Since the planning is so difficult in large-scale agile, there is resistance towards changing a plan or establishing it as a truly continuous activity.

*Unclear requirements*, one of the major challenges in large-scale agile planning, affect both estimation and prioritization. Our interviewees reported to be challenged to gain knowledge about the underlying user needs of a feature. This includes a potential lack of experience and knowledge about new features, starting with the area product owner.

Another challenge is the *unclear role of operational product owners* and our interviewees mention slightly different challenges in the two different products. In Product A, our interviewees expressed confusion about to what level teams should be involved in planning. In contrast, the responsibilities of operational product owners in Product B were clearer. They interacted quite naturally with informal leaders in the teams during planning, thus allowing involvement of the team, without distracting the team members too much from their development tasks. Accordingly, it is beneficial to share the plan with the whole team, without giving the team the formal responsibility to report on it.

Interviewees in both products agreed that balancing the *involvement of teams in large-scale agile planning* is challenging and that it is crucial to find a good process for their involvement. It is difficult for teams to engage in long-term planning beyond the next 18 months and while this might be necessary for large-scale agile development, our interviewees indicated that the teams do not benefit from participating in such long-term planning and do not understand why their participation should be necessary. While such

a long-term plan is interesting to program leaders, program managers, and to the product line, teams should focus on producing results and cut the lead-time.

Finally, our interviews reveal technical dependencies as well as dependencies between hardware and software as a planning challenge. If known, they represent constraints on planning. However, often such technical dependencies are hidden, leading to duplicate work or waiting time. While such waste cannot fully be avoided during large-scale agile planning, approaches to mitigate its impact are needed.

## 4.2 Contextual Challenges

In addition to the general ability to estimate, prioritize, and plan, our data also revealed a number of challenges with respect of the context of planning, including *work environment*, *team build-up*, and *team spirit*.

**Work Environment.** The studied company arranges the work environment generally as *open space*, where different teams sit to carry out their daily work. While this facilitates information flow between teams, one team's daily meetings can *disturb other teams*, as for example shown in the following quote from one of our interviews.

*"...we have scrum meetings in open office space [...]. You kind of get disturbed when other teams are having their scrum meeting in the open setting. It is better [if] every team has their different rooms."* [Dev.]

Another issue mentioned by our interviewees is that sometimes other team members disturb them by walking into their teams to ask for help. While this is important inter-team communication, *by-passing the operative product owners* can be problematic when it happens too often or on non-trivial issues.

**Team Build-Up.** Capabilities and special knowledge of teams are crucial resource constraints for planning. According to our interviewees operative product owners and program leaders have specific views on the *capabilities of the team*. This can lead to additional pressure on teams, when they are expected to develop more than what they are capable of. Our interviewees pointed out that these different views on teams capability can lead to frustration, when teams feel they cannot live up to the demand of the operative product owner and the operative product owner feels maybe she has promised something to the area product owners and the teams cannot deliver what she has promised.

A common practice is to *move team members* to other teams that require additional resources (such as special knowledge). Our interviewees indicate difficulties in finding candidates to be moved between teams. Also, they mention doubts about the effectivity of this practice, since the team member to be moved does not possess deep knowledge about the target team and their context. Our interviewees claimed that instead *competence broadening* of established teams should be emphasized to address missing team capabilities. Such a solution of course implies a longer lead-time and our interviewees pointed out that it is challenging for them to learn new roles when they already are experts in other required roles.

**Team Spirit.** Our interviewees described, how team spirit starts to grow when team members have worked together and functioned successfully for some time. As a result, removing or adding new members to the team might decrease the spirit that has already been established within the team. As discussed above, some times people have to be moved between teams to provide teams with required resources. According to our data, removing team members from an established team destabilizes its spirit, which has been built over the period of close teamwork. In addition to that, the team spirit in the receiving team can also be impacted. Teams and team members adopt agile methods in different ways and some engineers are less open for the agile mindset because they have used the traditional method for a very long time and are accustomed to it, which can further impact the spirit of agile teams.

### 4.3 Ceremonial Agreement

In addition to technical abilities and context of planning, we discovered themes in our interview data that affect the room between those two domains: Ceremonial agreement allows a group of agile teams and product owners to align planning abilities with the teams' context. Efficient and effective information flows are necessary for keeping every employee aware about important decisions. According to our interviewees, it is however challenging to share knowledge about decisions within the large development context due to a *lack of suitable information channels*. Our interviewees said that they do not get updated on what (other) teams are doing due to insufficient time. If a product owner is responsible for several teams and these teams have their stand-up meetings at the same time, he needs to decide. But also the opposite can happen, when a team or an operative product owner have to interact with two area product owners who are typically very busy. In this situation, it is impossible to have frequent face-to-face meetings with them, resulting in asynchronous communication via email and social media. Such communication is not as effective as face-to-face communication and can result in long response times.

Coordination meetings (such as scrum-of-scrum (SoS)) are a potential solution, but were also criticized as *boring* or *too short* by our interviewees. They pointed out that in most of the SoS meetings it is difficult to have a thorough discussion and arrive at a good conclusion. Most of the times they have to close the meetings when they get into interesting technical discussion. One OPO mentioned that such discussions in the meetings might not be interesting to all participants.

While it is important not to by-pass the operative product owner when communicating with the team, this also introduces some indirection, e.g. between release leaders and the team. This requires building trusted relationships between release engineer, operative product owner, and team. A lot of such communication is the consequence of *inadequate anatomy of features*, i.e. "*the relation between different features and parts of features*", as one of our interviewees put it. With other words, the way features are split up and assigned to sprint backlogs leads to dependencies between teams and creates the challenge of inter-team communication and coordination within the larger product portfolio. We found senior developers and product owners to rely on their *personal network* to coordinate across program boundaries:



*“...I have a colleague that works as [...] operative product owner in other program and we try to collaborate between the programs and to align features for the customers and user experience”. [OPO]*

Thus, we conclude that typical agile ceremonies are well adopted locally within teams, but challenges remain largely on the levels of inter-team and inter-product communication across a portfolio of products.

*“The biggest challenge I pick is the coordination of the feature portfolio, [...] on top of getting out features in our program fast and efficient, we need to collaborate on a portfolio basis to align the features over two programs”. [OPO]*

Again, our findings resonate with Sauer’s recommendation to facilitate team spirit with opportunities for informal exchange, such as coffee breaks [14]. Ceremonial agreements should support large-scale agile planning in similar ways.

## 5 Discussion and Conclusion

**Implications for Practice.** From a practical point of view, we see two main advantages: First of all, it is necessary to understand which different themes actually affect large scale agile planning. Too often too much effort is spend on shallow opinions, which then become the base for future actions. By having the seven different themes thoroughly understood, actions can be taken that lead towards true improvements. One example is the thorough understanding of the theme ‘Estimation Ability’ where a team needs a number of practices in place to be able to manage its velocity. Secondly, and likely even more important, it is necessary to understand how the different themes impact each other. For instance, once a team has understood its velocity and can estimate properly, they can be part of larger planning game among many teams, where in the end solid prioritizations can be made in favor of having one complete release ready in time that matches a market request. Understanding these correlations between the themes helps industry to organize improvement initiatives in a way where it becomes obvious when there will be a true contribution to the product development.

**Implications for Research.** Our main contribution in this paper is the model derived from our exploratory case study (summarized in Fig. 1). This model includes the insight that a large-scale agile organization’s ability of planning is not only depending on its teams’ abilities or skill, but also on the context in which those teams operate. Ceremonies and practices on inter-team and inter-product level are currently missing and invite further research. Our model gives an overview of key aspects of collaborative planning in large-scale agile development. And we hope that others find this overview useful to focus their research. In particular, we would encourage constructive research to provide improvement for one or several aspects. Our vision is a collection of best, or at least good, practices for each area in our model.

**Threats to Validity.** We carefully reviewed the codes and themes generated by our research method, to ensure that our results are correctly derived from our data. It was beneficial, that we could bring both industrial and academic expertise together in these

activities. Further, our qualitative investigation was carefully designed to align research method with research questions. Through reviews and pilot-interviews we made sure that participants of the study were able to understand and answer our questions, thus reducing the risk of misunderstanding and misinterpretation. Thus, we believe that we addressed internal and construct validity as well as reliability of the study in an adequate way. With our qualitative exploratory case study we did not aim for generalizability and external validity. Qualitative interviews and analysis are highly dependent on the researcher. To mitigate this threat to validity, we give a thorough description of context and procedures of our research in this paper. We are confident that repeating our study in a different, but comparable context will yield similar planning challenges of large-scale agile software development.

**Outlook.** When understanding themes and their correlation thoroughly, it is vital to get practices in place that embrace speed and responsiveness. These are the two key elements in agile development. Going forward, we see several different practices that could be further investigated: What methods can be used to improve team velocity? How can we organize work environments that facilitate a higher degree of responsiveness? Which ceremonies can be used to speed up the complete planning process? How can the planning process become more transparent? Are there any risks for planning too much? The potential of questions to continue to ask around large-scale agile planning is endless once the basic themes are understood and practiced to some level.

**Acknowledgements.** We thank our interviewees at Ericsson AB for their time and inspiring discussions. This work is partly funded by Software Center and Vinnova FFI project NGEA.

**Open Access.** This chapter is distributed under the terms of the Creative Commons Attribution-NonCommercial 4.0 International License (<http://creativecommons.org/licenses/by-nc/4.0/>), which permits any noncommercial use, duplication, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, a link is provided to the Creative Commons license and any changes made are indicated.

The images or other third party material in this chapter are included in the work's Creative Commons license, unless indicated otherwise in the credit line; if such material is not included in the work's Creative Commons license and the respective action is not permitted by statutory regulation, users will need to obtain permission from the license holder to duplicate, adapt or reproduce the material.

## References

1. Leffingwell, D.: *Scaling Software Agility: Best Practices for Large Enterprises*. Addison-Wesley Professional, Upper Saddle River (2011)
2. Larman, C., Vodde, B.: *Scaling Lean & Agile Development: Thinking and Organizational Tools for Large-Scale Scrum*. Addison-Wesley Professional, Upper Saddle River (2009)
3. Bass, J.M.: Scrum master activities: process tailoring in large enterprise projects. In: *Proceedings of 9th IEEE International Conference on Global Software Engineering* (2014)

4. Paasivaara, M., Lassenius, C.: Scaling scrum in a large distributed project. In: Proceedings of International Symposium on Empirical Software Engineering and Measurement (ESEM) (2011)
5. Paasivaara, M., Heikkilä, V.T., Lassenius, C.: Experiences in scaling the product owner role in large-scale globally distributed scrum. In: Proceedings of 7th IEEE International Conference on Global Software Engineering (ICGSE) (2012)
6. Abrahamsson, P., Salo, O., Ronkainen, J., Warsta, J.: Agile Software Development Methods: Review and Analysis (VTT publications), pp. 17–36 (2002)
7. Reifer, D.J., Maurer, F., Erdogmus, H.: Scaling agile methods. *IEEE Softw.* **20**(4), 12–14 (2001)
8. Cohen, D., Lindvall, M., Costa, P.: Agile software development. DACS SOAR Report, pp. 1–15 (2003)
9. Paasivaara, M., Durasiewicz, S., Lassenius, C.: Distributed agile development: using scrum in a large project. In: Proceedings of IEEE International Conference on Global Software Engineering (2008)
10. Dingsøy, T., Moe, N.B.: Research challenges in large-scale agile software development. *ACM SIGSOFT Software Engineering Notes* **38**, 38–39 (2013)
11. Braun, V., Clarke, V.: Using thematic analysis in psychology. *Qualitative Res. Psychol.* **3**, 86–93 (2006)
12. Creswell, J.W.: *Research Design: Qualitative, Quantitative, and Mixed Method Approaches*. Sage Publications, Thousand Oaks (2009)
13. Petersen, K., Wohlin, C.: A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case. *J. Syst. Softw.* **82**, 1479–1490 (2009)
14. Sauer, J.: Agile practices in offshore outsourcing—an analysis of published experiences. In: Proceedings of the 29th Information Systems Research Seminar in Scandinavia, IRIS, pp. 12–15 (2006)
15. Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. *Empirical Softw. Eng.* **14**, 131–164 (2009)
16. Hennink, M., Hutter, I., Bailey, A.: *Qualitative Research Methods*. Sage, Thousand Oaks (2010)
17. Beck, K.: *Extreme Programming Explained: Embrace Change*. Addison-Wesley, Boston (1999)
18. Schwaber, K.: *Agile Project Management With Scrum*. Microsoft Press, Redmond (2004)
19. Daneva, M., Van Der Veen, E., Amrit, C., Ghaisas, S., Sikkil, K., Kumar, R., et al.: Agile requirements prioritization in large-scale outsourced system projects: An empirical study. *J. Syst. Softw.* **86**, 1333–1353 (2013)
20. Larman, C., Vodde, B.: *Practices for Scaling Lean & Agile Development: Large, Multisite, and Offshore Product Development with Large-Scale Scrum*. Pearson Education, Boston (2010)