Inter-organizational Co-development with Scrum: Experiences and Lessons Learned from a Distributed Corporate Development Environment

Raoul Vallon, Stefan Strobl, Mario Bernhart, and Thomas Grechenig

Research Group for Industrial Software, Vienna University of Technology
Vienna, Austria
raoul.vallon@inso.tuwien.ac.at

Abstract. Distributed development within a single organization adds a lot of overhead to every software development process. When a second organization joins for co-development, complexity reaches the next level. This case study investigates an agile approach from a real world project involving two unaffiliated IT organizations that collaborate in a distributed development environment. Adaptations to the regular Scrum process are identified and evaluated over a six-month-long period of time. The evaluation involves a detailed problem root cause analysis and suggestions on what issues to act first. Key lessons learned include that team members of one Scrum team should not be distributed over several sites and that every site should have at least one Scrum master and one product owner.

Keywords: distributed development, agile development, Scrum, software development process, subcontracting, virtual teams.

1 Introduction

Agile development has gained widespread popularity over the last ten years in very different domains (e.g. embedded software projects [1], mobile application development [2] or aerospace [3]). It has been adopted by large companies such as Intel [4], Microsoft [5], Yahoo! [6] or SAP [7] and has thus found its way in multi-team and multi-site corporate environments. Although originally designed for collocated teams, related agile studies have reported the adaption of agile principles to e.g. a distributed Scrum [8], [9] or Extreme Programming (XP) [10] implementation in recent years.

Distributed development challenges one of the core strengths of Scrum: team members need to interact and communicate on a daily basis to form self-organizing teams and meet sprint goals. However, distributed environments complicate communication and coordination [11]. Technical tool support plays a bigger role in the process [12], [13] as well as knowledge management and transfer [14]. Consequently team members need to work harder to synchronize and meet sprint goals.

This case study strives to contribute to this field of research by investigating an agile distributed development approach based on Scrum. The process implementation involves two unaffiliated Austrian IT organizations, which are separated by about 300 kilometers. According to Kajko-Mattsson et al. [15], expected problem fields include communication, customer collaboration, trust, training and technical issues. We will further investigate the adaptations to Scrum and the compromises that need to be made, when two organizations with different corporate cultures join forces to develop software.

We defined the following research question:

RQ: How can agile development be applied to an inter-organizational, multi-site and multi-team development environment and what challenges, if any, emerge in this setting?

The rest of the paper is organized as follows. Section 2 describes the research settings and applied methods. Section 3 provides an observation of strengths and weaknesses in the process implementation. Section 4 conducts a problem root cause analysis. Section 5 discusses results including lessons learned, suggestions for practice and related work. Section 6 provides the conclusion.

2 Research Design

The case study covers a six-month-long period of time including evaluation and presentation of results. The nature of the case study is exploratory according to Yin's research on the application of case studies [16]. As such, it strives to identify problem areas in the field of agile distributed development and serves as a prelude to further follow-up studies. Findings of this exploratory study are put in context with related studies during the discussion of results.

2.1 Research Settings

Two unaffiliated organizations, the main supplier (MS) and the additional supplier (AS), collaborate to develop three software products that share a common codebase. Both suppliers have successfully applied regular Scrum before and chose to implement an adapted version of Scrum to better suit the needs of a distributed development environment. The two organizations develop at their own sites, separated by about 300 kilometers.

The MS is a large company whose IT department is involved in the development of the three software products. It is solely responsible for requirements engineering with all three customers and provides the bigger part of the development staff.

The AS is a medium-sized core software development company and a subcontractor to the MS for the development of the three products. It complements the MS's development with additional staff and know-how but has no contact with customers.

Table 1 shows the distribution of team members over the two suppliers. The MS has one product owner (PO) for each software product and three Scrum masters (SM) serving three teams. The AS does neither have a PO nor a SM on site.

Table 1. Distribution of team members over the two suppliers

Co-Developers	Dev	Test	SM	PO	Sum
Main Supplier (MS)	11	3	3	3	20
Additional Supplier (AS)	8	2	0	0	10
Overall	19	5	3	3	30

2.2 Research Method

The research is divided into three phases.

Observations. One of the authors examined the Scrum implementation in use as an external observer. As such, he took part in various meetings and conducted interviews with members of all roles (product owner, Scrum master, developer, tester). The interviews lasted from 20 to 45 minutes and have been audio-recorded. He took field notes, pictures and collected planning sheets and meeting minutes. He has been granted read-only access to several electronic tools involved such as the issue tracking system. This phase lasted for three months.

Case Analysis. After the observation phase the collected data was analyzed. The authors extracted problems from the following sources: retrospective meetings, interviews, field notes, meeting minutes and the project documentation. Problems were categorized in problem clusters and root causes suggested in a problem root cause analysis. The approach was top-down, i.e. most prominent problem clusters were analysed first according to the authors' evaluation. This phase lasted for two months.

Presentation of Results. The last phase involved a presentation and discussion of results with team members including lessons learned, suggestions for practice and related studies. This was the concluding step in the last month.

3 Observation Phase

The following observations summarize the different aspects of the Scrum implementation applied in the case study including strengths and weaknesses.

Formation of Scrum Teams. Three Scrum teams have been formed across all products and based on logical requirement areas. Figure 1 shows the distribution of team members. The product owner and the Scrum master roles are both on the MS's site. The AS complements the MS with additional developers and testers (QA) but has no official Scrum roles. However, two developers have emerged as unofficial Scrum masters for the AS. They care most for the process implementation and discuss impediments with the MS. One of these unofficial Scrum masters travels to the MS's site once a week for face to face updates and discussions.

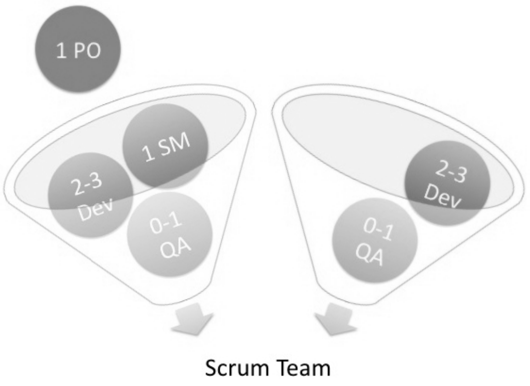

Fig. 1. Distribution of Scrum team members: the AS joins the MS's Scrum team with developers and QA but has no official Scrum roles

Each Scrum team holds a daily video conference meeting, where respective team members of the MS and AS participate. Additionally a Scrum of Scrums (SoS) meeting is established for inter-team communication, as pictured in figure 2.

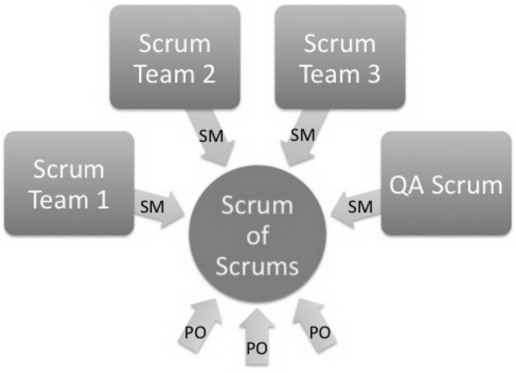

Fig. 2. Scrum of Scrums (SoS) is held at the MS only. Testers of all three Scrum teams form a virtual QA Scrum team and also participate in the SoS.

It is held daily at the MS's site. Since the AS does not have official Scrum roles, the MS handles all inter-team coordination. Testers (QA) are assigned to the three Scrum teams, but hold an additional daily meeting to stay synchronized and also send a representative to the SoS. Product owners also participate to evaluate the progress.

Two-Tiered Planning Process. Planning covers one month, i.e. two sprints. It is a two-tiered process: at first, planning is done at the MS's site with one of the two unofficial Scrum masters of the AS present. The Scrum teams decide, which of the prioritized user stories in the product backlogs they want to implement in the next two sprints.

The second level planning continues at the AS's site: The unofficial Scrum master returns from the MS with pre-estimated (via planning poker [17]) user stories for the AS. Team members volunteer for certain tasks until all tasks are assigned. When a team member accepts a task, it adjusts the original estimation of the MS to his own. One of the unofficial Scrum masters updates a planning spreadsheet during the meeting and shares it with the MS afterwards.

Joint Sprint Review. The sprint review is held after each sprint. It is primarily held at the MS, but the AS joins via video conference. Additionally, one of the AS's Scrum masters is present at the MS's site to represent the AS in person as well. The rest of the AS's team is mainly observing the review, but can raise questions or concerns when necessary. The review consists of feature demonstrations and discussions about different areas of the current product increments and takes about two hours.

Joint Retrospective. The sprint retrospective is held monthly after two sprints with the same setup as the sprint review. The retrospective is divided into six steps (the AS's on-site representative conducts the steps on behalf of his colleagues):

1. Individual evaluation of the last month from good to bad on a 15-part scale. Each team member may put one point on the scale drawn on paper.
2. Evaluation and discussion of the measures taken against impediments since the last retrospective.
3. Every participant writes three remarks (either positive or negative) on paper and puts them on the flipchart, shortly presenting each.
4. The individual remarks from step 3 are clustered to topics.
5. Every participant has three points that can be assigned to one or more of the clustered topics according to his personal weighting.
6. Measures for the top three topics are discussed that will be implemented in the next two sprints.

Product and Sprint Backlog. Each product owner maintains a product backlog on the MS' site for his product. At the time of the observation phase the AS did not have access to the product backlog, but worked with the sprint backlog only (planning spreadsheet from the two-tiered planning process).

Scrum Board. Both the MS and the AS are using paper Scrum boards. Each Scrum team operates one board. Since the two suppliers are based at different locations, six boards would be needed, but the AS currently only uses one general board on his site covering all three teams. The workflow on the board is defined as: *User Stories*, *TO DOs*, *In Progress*, *Review* and *Done*.

The first column *User Stories* contains user stories from the backlog. Sticky notes of the same color are used to break the user stories into smaller tasks that run through the remaining workflow. The column *Review* denotes the tasks being reviewed and tested by a colleague (at any supplier's site). Tasks in *Done* are production-ready. Tasks on the Scrum board are also marked with the issue tracking number of the electronic tool in use. The Scrum boards of both suppliers are synchronized every day during the daily Scrums (for each team).

Burndown Chart. The burndown charts are drawn and updated on paper at the MS's site only (one per team). The AS does not operate one on his own, but the MS includes the AS's tasks in his chart.

Behavior Driven Development. The two suppliers develop software using *behavior driven development (BDD)* [18], which is an extension to *test driven development (TDD)* [19]. The goal is to define the software's behavior in terms of human readable, but executable acceptance criteria [18]:

```
Given some initial context,
When an event occurs,
Then ensure some outcomes.
```

These acceptance criteria can be automated to test the correct behavior of the software. They should be understandable to the customer yet precise enough to be executable. BDD also helps provide a common language and reference point for stakeholders, business analysts, developers and testers.

Means of Communication. The main means of communication between the two suppliers are joint meetings via video conference and telephone calls. Individual concerns are discussed in emails, instant messaging and screen sharing sessions.

3.1 Retrospective

In the three retrospective meetings during the observation phase, issues overweighed strengths by far due to the complex development environment. Named strengths were *improved communication and collaboration* in general and *continuous improvement*. Team members identified the following drivers for improvement:

- Willingness and commitment to change and improve
- Good working atmosphere and employee attitude
- Highly motivated people
- Team work

The list of problems taken from retrospective meetings is notably longer. Both team members of the MS and AS reported to suffer from **constant stress in the two-week sprint** due to the following reasons:

- Workload too high in relation to available staff
- Planning delay in general and also between the two suppliers
- Too little time to follow BDD workflow in a two-week sprint

Late planning was reported since inter-organizational planning was frequently not ready until a few days into the sprint iteration. This made it very hard for team members to reach sprint goals. The **BDD workflow** introduced a lot of overhead. Testers constantly struggled to finish automation of BDD scenarios within the Sprint which resulted in broken test cases and thus **bad code quality**. Problems with the speed of **remote access for the AS** arose, which slowed down co-development. Minor issues regarding the **quality of use cases** were also reported.

3.2 Interviews

Three prominent issues have been identified from one on one semi-structured interviews that have been stressed by all interviewees. One of these issues, **overhead of communication and coordination**, addressed the inadequate quality of video conferences, especially with larger groups (joint sprint review/retrospective). A **lack of electronic tool support** has also been criticized, especially by the AS. The AS did not have access to the main paper Scrum boards and burndown charts at the MS's site and progress was synchronized mostly during daily Scrum meetings. **Two-tiered sprint planning** put pressure on team members' commitment since planning took too long and was frequently not ready at the beginning of new sprint iterations.

3.3 Summary

Table 2 provides an overview of observed problems and their weighting by team members during retrospective meetings. After all of the interviews were conducted, each interviewee was asked to select the most prominent problem out of three problems that arose in all interviews. The ranking is also shown in table 2.

Table 2. Observed problems in the case study

Source	Problems	Weighting by team members
Retrospectives	High Stress-Level	30,7%
	Late Planning	25,8%
	BDD Workflow	12,9%
	Code Quality	12,9%
	Remote Access for AS	9,7%
	Use Cases	8,0%
Interviews	Communication and Coordination Overhead	1st
	Lack of Tool-Support	2nd
	Two-Tiered Sprint Planning	3rd

4 Case Analysis

After the observation phase, the data collected was analyzed. Problems were identified and clustered from different sources: retrospective meetings, interviews, field notes, meeting minutes and the project documentation. The result was eight problem clusters with the following top-down prioritization: distributed development, transparency, commitment, planning, estimation, predictability, self-organizing teams and tools. The problem clusters have been analyzed for two months. Table 3 shows the result of the analysis: problem clusters and corresponding identified root causes in the case study.

Table 3. Problem clusters and identified root causes

Problem Clusters	Identified Root Causes
Distributed Development	No Official Scrum Roles at the AS
	Joint Estimation and Planning
	Inter-Company Distribution of Team Members
Transparency	Suppliers not Collocated
	Communication Issues
	Little Documentation
	No Overview over All Teams
Commitment	Commitment Fails with Insufficient Planning
	Commitment Fails with Late Planning
	Commitment Fails with Frequent Changes
	Little Respect for Iterations
Planning	Late Actual Beginning of Sprint
	Little Participation of AS
	Little Information for AS
Estimation	User Story Estimation in Hours
	Pre-estimations by MS
Predictability	No Proper Sprint Velocity
	Further Impediments for Better Predictability
Self-Organizing Teams	Tasks Assigned to Team Members
	Estimations Based on Individuals
	Cross-Team Working Agreements
Tools	Tools Lack Scrum Compatibility
	Limited Remote Access for AS
	Paper Scrum Board and Burndown Chart

Distributed Development. All three Scrum teams are staffed by members of both suppliers, yet all product owners and Scrum masters are on the MS's site. Nevertheless, two of the AS's team members have emerged that do more coordination work than their colleagues. They care more for the Scrum process than others (Scrum master) and travel to the MS to attend meetings and discuss user stories in person (product owner). The team members on the AS's site are 10 people distributed over three different Scrum teams. It is very hard to remain self-organizing and in compliance with the Scrum process, when contact to the remaining team members is hard to establish and no role is officially assigned to look after the process at the AS's site.

This poses a big problem for the AS, as these two to three team members are separated from the rest of the MS-based team. As a result, the AS has formed a virtual team to manage his own resources with a single paper Scrum board covering all three teams. The follow-up planning session is also held for the whole AS's virtual team, including members of all three real Scrum teams.

Transparency is a big issue between the two suppliers due to the physical distance of 300 kilometers. The whole process becomes more complex and less transparent. Low quality video conferences and little available documentation further handicap communication and coordination. There is no high level overview of the progress of all three teams available to everyone since Scrum boards and burndown charts are drawn on paper.

Commitment is hard to achieve with late planning and frequent changes within the sprint iteration. The teams cannot commit to sprint goals when the user stories are not properly and timely specified. As a result, estimations are not reliable.

Planning and Estimation. The MS pre-estimates user stories and uses this estimation as a basis for planning. The AS is thus not adequately involved in the planning process apart from updating the estimations of the MS (for his own user stories only). Planning is often not ready until a few days into the sprint, which causes delays for both suppliers. Estimation is done in hours. This does not represent complexity well because different people need different amounts of time to work on a user story.

Predictability. Sprint velocity cannot be properly measured because the MS runs a paper burndown chart that is based solely on tasks (derived from user stories). The only available ratio is tasks per sprint, which does not represent any complexity because it does not take into account hours (or story points). Further impediments to a better predictability are a varying understanding of the BDD workflow among team members and code quality issues.

Self-organizing Teams. Two developers emerged at the AS's site that do more coordination work and impediment handling than others. The distributed environment complicates coordination between teams and it is hard for the AS to efficiently complement the MS-based teams. Moreover, cross-team working agreements regarding the BDD workflow need to be elaborated and agreed upon to reduce interdependency issues.

Tools. The electronic tools in use all lack Scrum support, which prevents a proper process implementation. There are currently four paper Scrum boards in use, three at the MS's site for each team and a combined one at the AS's. These are cumbersome to synchronize, which slows down the tracking of other teams' progress. The burndown charts are also drawn on paper and only available to the MS.

5 Discussion

The six-month-long case study involved two suppliers MS and AS from unaffiliated organizations, which joined forces to co-develop three software products. The research question was "*How can agile development be applied to an inter-organizational, multi-site and multi-team development environment and what challenges, if any, emerge in this setting?*

The following adaptations to the Scrum process have been made for the case study's setting:

- Two unofficial Scrum master-like roles emerged at the AS that frequently paid visits to the 300-kilometers-away MS to improve the flow of information
- Three Scrum teams have been formed, each consisting of members from both development partners
- Joint daily Scrum/two-week review and monthly retrospective meetings are held via video conference calls
- Scrum of Scrums is held daily at the MS's site without participation of the AS
- Two-tiered planning process (MS first, AS second) in use covering two sprints

The analysis showed that finding a working Scrum implementation is indeed very challenging in an inter-organizational distributed development setting. Eight problem clusters were identified as illustrated in figure 3. The relation between problem clusters may not be as linear in real-world projects, but it serves as an illustration of underlying constraints. It should also be regarded as an impulse on what problems to act first: the suggested approach is bottom-up starting with enabling truly self-organizing teams.

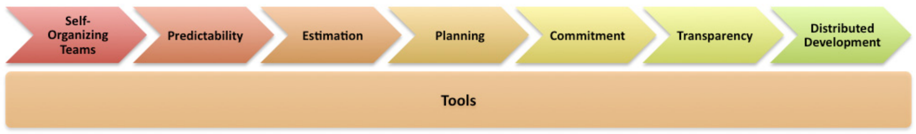

Fig. 3. Proposed identified relation of problem clusters. Problems should be solved bottom-up from self-organizing teams to distributed development.

Self-organizing teams are one of the central components of Scrum and need to be established first. *Predictability* evolves when long-lived self-organizing teams are allowed to work in sprint iterations without outside interference. *Estimation* and *planning* can only be accurate once predictability is reliable. The case study shows that *commitment* cannot be achieved without timely planning. Providing *transparency* is one of Scrum's highest goals, i.e. making impediments visible to everyone. All the precedent problem categories need to be solved before transparency can be achieved.

Distributed development is the central problem in this case study, since the two suppliers are not collocated. Collaboration can only be improved by solving the other problem categories first. Choosing the right *tools* for the specific project setting supports the whole value stream and has even greater impact in distributed development environments as team members need to rely more heavily on electronic means of communication.

The major problem in the Scrum implementation was that the process was focused on the MS. This observation is supported by numerous identified root causes during the problem root cause analysis:

- No official Scrum roles at the AS
- Little participation of AS
- Little information for AS
- Paper Scrum boards and burndown charts
- Pre-estimations by MS
- Limited remote access for AS

Due to these reasons, planning and commitment frequently failed during the observation phase in the case study. During retrospective meetings the general consensus was that communication and coordination between the suppliers is improving, but one on one interviews still disclosed many problems.

The MS is the main contractor in this project environment. Greater involvement of the AS could lead to a more efficient development output. Scrum does not work well in a hierarchical setting as the formation of self-organizing teams is denied and transparency is decreased.

5.1 Lessons Learned

Inter-organizational Co-development Adds Another Layer of Complexity. The case study shows that co-development between unaffiliated organizations adds new complexity and challenges to overcome. The reasons are often organizations varying in size and corporate culture. The introduction of hierarchies has no space in agile development. The case study shows that overall transparency and thus efficiency will decrease. In terms of development output, distributed teams cannot compete with collocated teams on average due to the complexities involved. Hence the decision to distribute development should be considered carefully.

Increased Effort for Self-organizing Teams. Both suppliers have run regular Scrum before. Retrospective meetings and observations showed that the distribution of development across two suppliers complicated software development. The level and willingness of cooperation between team members determines success or failure. In general, it is harder for teams to remain self-organizing as more effort is needed to synchronize with distant team members.

Organizational Change Takes Time. The larger the organization, the harder it is to introduce changes. This could especially be observed with the MS, where changes took long to be realized compared to the AS which is smaller in size. Compromises had to be made to deal with organizational impediments such as the switch to paper boards.

Beware of a Superficial Scrum Adoption. The number one organizational impediment to a successful adoption of agile principles is silver bullet thinking and superficial adoption [20]. The case study showed that although distributed development caused many problems, underneath many "regular" Scrum values have not been met, such as self-organizing teams and the respect for iterations.

5.2 Suggestions for Practice

A lot of coordination and synchronization overhead was introduced by having multi-site Scrum teams. We suggest **forming single-site Scrum teams only**. For this case study's setting an additional Scrum team on the AS's site can be formed including a Scrum master and a product owner instead of having three multi-site teams. The **on-site Scrum master and product owner** also enforce an **equal involvement of all sites** in the distributed Scrum process. Additionally **decent electronic tool support for the Scrum process is essential** in a distributed environment for inter-team coordination.

These measures help increase transparency and improve overall development output. The case study showed that the appreciation of agile core values such as the respect for iterations is of major importance especially in distributed development environments.

5.3 Related Studies

The suggestion to **form single-site Scrum teams** aligns with one of the best practices of the Scrum Alliance: form distributed but isolated Scrum teams that are linked through the Scrum of Scrums [21]. However, Sutherland et al. provides a success story in [21] stating that distributed integrated teams (over two sites) are more efficient than the suggested best practice. Our exploratory study shows many identified problems with the latter approach. Hence, we suggest implementing the Scrum Alliance's best practice, especially if team members are not agile experts. Vax et al. also conclude in [22] that you need the right expertise and team for distributed Scrum.

Penttinen et al. propose guidelines in [23] for three types of subcontracting teams in Scrum: sub-contractor team (team with only sub-contractor members), mixed team (an on-site mixed team) and a virtual team (a multi-site mixed team). In our case study we had three virtual teams. Penttinen et al. also conclude that a virtual team is the most complex option and most of the time a temporary one.

Instead of an **on-site Scrum master or product owner**, Paasivaara et al. mention the possibility of having an "ambassador/rotating guru" in [8], who is sent to other sites for a longer period of time. This measure serves as a compromise between a full on-site Scrum master/product owner and the short-term visits conducted in the case study at hand.

In related publications we can see a growing interest in bringing agile to (globally) distributed software development [24], [25]. One of the conclusions over several case studies reviewed by Hossain et al. in [24] is that Scrum needs to be extended to work in a distributed setting, which has also been shown in our case study.

5.4 Limitations

Since this is a single case study, generalizability of results is limited. One of the authors took the role of an external observer to minimize bias. As such, he was not part of the team and thus was not able to fully capture each detail of daily work.

6 Conclusion

This case study investigated a Scrum-based agile approach to distributed development between two unaffiliated organizations. We identified that the prominent problem was that the developing partners formed an unequal partnership. The MS provided two thirds of the staff involved. The AS joined as a subcontractor with developers and testers but had no Scrum roles on site. Joint retrospective meetings showed that the stress level was very high for both development partners. The main reason was a weak flow of information between the MS and the AS, which resulted in frequent issues with planning and estimations. Although the coordination and communication improved over time, it has still been the main issue in most interviews conducted.

The fact that two unaffiliated organizations joined forces to develop a software product added a new layer of complexity to distributed development. The Scrum adaptations included moving most regular Scrum meetings to video conference ones, but the process implementation was strongly focused on the MS: The Scrum of Scrums was held in person at the MS's site only and the AS did not have any official Scrum roles. The paper Scrum board and burndown charts were also based at the MS's site which decreased transparency for the AS.

We suggest the formation of single-site self-organizing teams instead of multi-site ones. Scrum masters and product owners should be present on all sites to ensure an equal involvement of all developing parties in the process and improve the flow of information. The case study further showed that an extensive electronic tool support is crucial to the self-organization of teams in a distributed development environment.

References

1. Xie, M., Shen, M., Rong, G., Shao, D.: Empirical Studies of Embedded Software Development Using Agile Methods: a Systematic Review. In: 2nd International Workshop on Evidential Assessment of Software Technologies, pp. 21–26. ACM, New York (2012)
2. Scharff, C., Verma, R.: Scrum to Support Mobile Application Development Projects in a Just-in-time Learning Context. In: 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering, pp. 25–31. ACM, New York (2010)

3. Vander Leest, S.H., Buter, A.: Escape the waterfall: Agile for aerospace. In: 28th Digital Avionics Systems Conference, pp. 6.D.3-1–6.D.3-16 (2009)
4. Chen, J.Q., Dien, P., Wang, B., Vogel, D.R.: Light-Weight Development Method: A Case Study. In: 2007 International Conference on Service Systems and Service Management, pp. 1–6 (2007)
5. Begel, A., Nagappan, N.: Usage and Perceptions of Agile Software Development in an Industrial Context: An Exploratory Study. In: 1st International Symposium on Empirical Software Engineering and Measurement, pp. 255–264 (2007)
6. Chung, M.-W., Drummond, B.: Agile at Yahoo! From the Trenches. In: 2009 Agile Conference, pp. 113–118. IEEE Computer Society, Washington, DC (2009)
7. Schnitter, J., Mackert, O.: Large-Scale Agile Software Development at SAP AG. In: Maciaszek, L.A., Loucopoulos, P. (eds.) ENASE 2010. CCIS, vol. 230, pp. 209–220. Springer, Heidelberg (2011)
8. Paasivaara, M., Durasiewicz, S., Lassenius, C.: Using Scrum in Distributed Agile Development: A Multiple Case Study. In: 4th International Conference on Global Software Engineering, pp. 195–204 (2009)
9. Bannerman, P.L., Hossain, E., Jeffery, R.: Scrum Practice Mitigation of Global Software Development Coordination Challenges: A Distinctive Advantage? In: 45th Hawaii International Conference on System Science, pp. 5309–5318 (2012)
10. Hildenbrand, T., Geisser, M., Kude, T., Bruch, D., Acker, T.: Agile Methodologies for Distributed Collaborative Development of Enterprise Applications. In: 2008 International Conference on Complex, Intelligent and Software Intensive Systems, pp. 540–545 (2008)
11. Korkala, M., Abrahamsson, P.: Communication in Distributed Agile Development: A Case Study. In: 33rd EUROMICRO Conference on Software Engineering and Advanced Applications, pp. 203–210 (2007)
12. Dullemond, K., van Gameren, B., van Solingen, R.: How Technological Support Can Enable Advantages of Agile Software Development in a GSE Setting. In: 4th International Conference on Global Software Engineering, pp. 143–152 (2009)
13. Niinimäki, T.: Face-to-face, Email and Instant Messaging in Distributed Agile Software Development Project. In: 6th International Conference on Global Software Engineering Workshop, pp. 78–84 (2011)
14. Dorairaj, S., Noble, J., Malik, P.: Knowledge Management in Distributed Agile Software Development. In: 2012 Agile Conference, pp. 63–73 (2012)
15. Kajko-Mattsson, M., Azizyan, G., Magarian, M.K.: Classes of Distributed Agile Development Problems. In: 2010 Agile Conference, pp. 51–58 (2010)
16. Yin, R.K.: Applications of Case Study Research (Applied Social Research Methods). Sage Publications (2011)
17. Grenning, J.: Planning Poker or How to Avoid Analysis Paralysis While Release Planning (2002), `http://renaissancesoftware.net/files/articles/PlanningPoker-v1.1.pdf`
18. North, D.: Behavior Modification. The evolution of behavior-driven development. Better Software Magazine (March 2006)
19. Beck, K.: Test Driven Development By Example. Addison-Wesley Professional (2003)
20. Larman, C., Vodde, B.: Scaling Lean & Agile Development. Thinking and Organizational Tools for Large-Scale Scrum. Addison-Wesley, Boston (2009)
21. Sutherland, J., Viktorov, A., Blount, J., Puntikov, N.: Distributed Scrum: Agile Project Management with Outsourced Development Teams. In: 40th Hawaii International Conference on System Sciences, p. 274a (2007)
22. Vax, M., Michaud, S.: Distributed Agile: Growing a Practice Together. In: 2008 Agile Conference, pp. 310–314 (2008)

23. Penttinen, M., Mikkonen, T.: Subcontracting for Scrum Teams: Experiences and Guidelines from a Large Development Organization. In: 7th International Conference on Global Software Engineering, pp. 195–199 (2012)
24. Hossain, E., Ali Babar, M., Paik, H.: Using Scrum in Global Software Development: A Systematic Literature Review. In: 4th International Conference on Global Software Engineering, pp. 175–184 (2009)
25. Jalali, S., Wohlin, C.: Agile Practices in Global Software Engineering – A Systematic Map. In: 5th International Conference on Global Software Engineering, pp. 45–54 (2010)