Large-Scale Agile Software Development at SAP AG

Joachim Schnitter and Olaf Mackert

SAP AG, Dietmar-Hopp-Allee 16, 69190 Walldorf, Germany
{j.schnitter,olaf.mackert}@sap.com

Abstract. In an ongoing change process SAP AG has managed to move the software development processes from a waterfall-like approach to agile methodologies. We outline how Scrum was introduced to implement a lean development style as well as the model chosen to scale Scrum up to large product development projects. The change affected about 18.000 developers in 12 global locations. This paper is an extended and revised version of an earlier publication [15]. It includes recent findings.

1 Introduction

Agile methodologies have proven to offer many benefits when developing user-centric software. Early findings about how to optimize product development led to iterative processes stressing prototyping and early feedback [19]. Agile project management methods, e. g. Scrum [16,17], take into account that product development is essentially a learning process. Long-term planning based on product requirements, which appear rock-solid but are in reality merely educated guesses, is substituted by an iterative approach which focuses on early delivery of partial functionality. In combination with regular feedback from users and stakeholders a software development project converges into something useful without the exact goal being known beforehand. Strict prioritizing of all requirements ensures that risks remain low. Even a project that was cancelled because of time and budget constraints may nevertheless have delivered a useful result if the essential requirements were implemented by the time the project was cancelled.

Scrum is known to work well in teams of up to around 10 people programming for a customer on the basis of a project contract. How well can Scrum perform in a global software company with thousands of developers in various locations working on a few dozen highly complex software products? This paper describes some of the experiences gained during the change process at SAP AG, the world's leading producer of enterprise software, transitioning from a waterfall-like overall process model to an agile development model that incorporates the values of lean development and production.

In the following section we will outline the situation at SAP before the introducton of agile development. The third section describes the activities during and results of the pilot phase. The fourth section gives a brief overview of process tailoring before introducing Scrum throughout the development organizations. In the fifth section we describe the steps taken to introduce lean development. In the sixth section the model

L.A. Maciaszek and P. Loucopoulos (Eds.): ENASE 2010, CCIS 230, pp. 209–220, 2011.

chosen to scale Scrum up to very large development projects is described. The final section summarizes the results and observations.

2 SAP Before Lean Development and Scrum

SAP, founded in 1972, offers a portfolio of enterprise applications around its flagship, the SAP Business Suite. Many of these products are aimed at large companies, although products for small and medium businesses have been available for some time or are in the making. With the acquisition of Business Objects SA in 2008, SAP gained access to user-centric analytics products which have since been fully integrated into SAP's traditional product line. In 2010 SAP started shipping new applications and platforms based on in-memory database technology which is also to be included in the products of another recent acquisition of SAP, Sybase Inc.

In its first two decades SAP developed software for and together with its customers. This close co-operation helped the company focus on business essentials, keep pace with business and technology trends. It eventually led to the fully developed client-server System R/3 which supported all critical business functions for every major industry. Developers had regular contact with customers and users in different roles, not only as programmers but also as consultants, trainers, and support engineers. The success of R/3 reshaped the market for business applications. Newer customers, though, expected that R/3 was a finished, standardized product they could easily install, configure and run. At the same time many of them expected full support of their individual business processes. Demand shifted from more business functionality to a higher degree of configurability, easier lifecycle management, and solutions covering only parts of the R/3 application suite with more detail and specificity.

Within a few years SAP's development organization grew from about 2,000 to over 18,000 employees in various functions. In earlier days developers were expected to be generalists, but by the time growth started slowing most developers had become experts on a particular technology or application area. Specialization increased in parallel with division of labour. By now there were experts working on requirements roll-in, other experts working on detailed requirements specifications, yet others laying out the basic functions and algorithms, programming, testing, and so on. By 2001 it had become clear that issues with communication among the specialized teams, departments, organizations were becoming a major cause of reduced product quality on many levels. To overcome these problems SAP introduced a development process framework to co-ordinate development efforts throughout all involved organizations. Involvement and responsibilities were assigned to departments according to their primary activities and skills overall leading to a waterfall-like process model with handover procedures among the involved parties. This process model was traversed once per year by most development departments which resulted in new products or major product versions each year.

When agile methodologies gained broader public interest around 2004 some teams at SAP started to experiment with Extreme Programming and Scrum. This approach was not feasible for regular product development teams because they had no longer easy access to customers and users. Therefore agile project management practices were first used in prototype development with strong customer involvement.

3 Pilot Phase

In 2006 SAP formed a small expert team of people experienced in agile development practices. This team began to educate and support development teams interested in Scrum. Within two and a half years it was possible to run a substantial number of projects according to the Scrum methodology. During this time the expert team gained important insight into applicability, limiting factors, and resistance within the organization.

Pilot teams were chosen by organizations according to their own interest. Since some of the early teams displayed scepticism due to personal attitude, each project team had to approve its participation in the pilot program unanimously.

3.1 Education

Scrum education for pilot teams consisted of the following elements:

1. A two-hour introduction into motivation and practices of Scrum for development managers.
2. A six-hour training for the development team, Scrum master, and product owner. This included an in-depth analysis of project setup, risks, and interdependencies with other teams.
3. A four-hour in-depth training for Scrum masters and product owners.

Each person in the expert team supported a number of pilot teams for some months as a Scrum mentor. This support included participation in all sprint planning and review meetings, retrospectives, help with tools, and advocating Scrum before development management. Additionally mentors tried to track and resolve all issues the teams had either internally or with other teams, organizations, and management.

After eight months of direct team support the Scrum expert team started offering trainings for future Scrum mentors. Mentor training took one week and included in-depth Scrum knowledge, team dynamics, and selected areas of software engineering, e. g. requirements management. A prerequisite for the mentor training was experience with Scrum either as a Scrum master or product owner. Training included Scrum master certification by the Scrum Alliance. The teaching team consisted of the Scrum expert team, a psychologist, and a practitioner from the Scrum Alliance.

After one year there were 25 mentors available. By September 2008 these mentors had supported about 120 projects in 10 global development locations.

3.2 Project Management Tools

When the Scrum expert team started its work, few project management software systems were available which supported Scrum. They all lacked sub-project support, hierarchical backlog management, and interfaces for inclusion into SAP's toolchain, e. g. to the requirements database, issue tracker, and test management systems. Therefore SAP decided to extend its own product for project management, SAP cProjects, to support Scrum projects. This extension project was also a showcase for Scrum with direct involvement of users because the system was to be used internally. A member of the Scrum expert team took on the role of the product owner.

3.3 Lessons Learned

A wiki was used to manage basic information and observations. For each pilot team the following data were collected by the Scrum mentors: scope of the project, duration, product owner, scrum master, dependencies with other teams, locality, critical decisions and impediments, and other observations. These data were used to select and communicate best practices, create a network of Scrum masters and product owners, and set up a round table of experienced practicioners. In-depth interviews with Scrum masters, product owners, and team members provided valuable information and suggestions for improvement.

Scrum gives teams more power to make decisions concerning development speed and quality. These aspects led to wide acceptance among teams. On the other hand many individuals felt that they were being monitored exessively as a result of the high degree of interaction both within the team and with external individuals and organizations. The predominant arguments against Scrum were:

1. Team member: "Scrum forces me to report to the team every failure and impediment. This puts me under pressure."
2. Team member: "I work more effectively when working alone. Daily Scrum meetings force me to interrupt my work."
3. Team member: "My manager might get a wrong impression of my work in case someone reports how slow my work progresses."
4. Development manager: "I have no insight into the team's work. Appearently nobody monitors what is going on."
5. Other teams: "We cannot rely on teams which do not deliver according to an agreed-upon schedule."

Most resistence of team members was removed by pointing out that the transparency of activity within the team was necessary because it enabled the team to manage its work independently. In order to be released from management's scrutinity the team has to monitor itself. Sceptics from management were invited to sprint planning and review meetings. They were regularly impressed by the teams' skills in planning all activities with great detail, focusing on risk management and high product quality.

What could not be mitigated were issues with teams dependent on "unreliable" Scrum teams, at least if the Scrum teams were isolated in a non-Scrum environment. Scrum teams working jointly on a particular software product rarely had complaints about unreliable delivery because they were familiar with the Scrum methodology and figured out how to get their requirements added to other teams' backlogs and with the right priorities.

Globally distributed teams often successfully applied Scrum although we were able to identify a number of problem areas which increased the reluctance to set up new distributed project teams:

1. Project kick-off meetings with all team members brought together in one location were rarely performed but were considered to be essential.
2. Neither telephone nor video conferencing systems could produce the feeling of proximity which was necessary to maintain team spirit.

3. Oral communication was burdensome because in every location English was spoken with a local accent.
4. Timezone differences of more than three hours made it difficult to agree on a common time for the daily Scrum meeting.
5. Multicultural teams typically broke up within two months mainly because the role of the Scrum master is interpreted very differently among European, Asian, and American people. A Scrum master from one culture found it hard to meet the expections of team members from another culture.

Many issues were not directly related to Scrum usage but became visible in this context. Teams tended to relate problems to Scrum usage, but deeper analysis showed that these issues had existed in the same teams before or could be observed in non-Scrum teams as well. This showed the high potential of Scrum to expose problem areas which otherwise would have gone unidentified or had been hidden intentionally.

3.4 Other Issues

Around the same time some problems surfaced which were not connected to Scrum usage but needed to be resolved before moving to Scrum on a large scale:

1. Communication among product management, software architects, development teams, and quality assurance was not strong enough to ensure a common understanding of development goals. Product management often was unable to communicate product requirements to software engineers. Languages and notations differed significantly.
2. The role of software architecture (high-level design) was not yet well understood. This led to increased dependencies among products and components, e. g. because componentization was weak and code reuse was highly valued. Another result was the lack of conceptual preparation for the development teams for whom there were too many obstacles to harmonize on interfaces and adhere to delivery timelines.

To overcome both problems a team was formed to provide education on software architecture for developers, software architects, and product management. A simple modeling language was created by combining selected Unified Modeling Language (UML) elements with diagram types from the Fundamental Modeling Concepts (FMC) notation [6,7]. FMC block diagrams in particular have proven to be an immense help to communicate technical concepts to customers, product managers, and developers. In parallel SAP revised its internal quality standards and added rules for component separation and usage.

4 Development Process Reform, Round One

A major revision of the standard development process took place while the Scrum expert team was still running pilot projects. Early experiences with Scrum came to the attention of management who ordered a reform of the development process framework to involve customers and users more directly. The “reform” team which included the members of the Scrum expert team defined three process models for different product development types.

Improvement. This process model is used to improve existing products and for minor extensions. In the context of business software this can mean e. g. adaptions to changed legal requirements or newer technical standards. An important criteria for this process model is that the project requirements include few changes to implemented business processes, only minor changes to the user interface, and that no newer technologies need to be added.

New Product. This model is chosen for new products or major product extensions. It includes Scrum as the project management method. Customer and user involvement is secured by appropriate contracts, and a minimum number of customers has to be involved. A calculated business case must exist as a basis for a development decision.

Research. This process model is used basically for research and prototyping projects. These projects are mostly free to chose whatever tools and methods they wish.

An important result of this approach was that it increased awareness of the need to tailor the process model according to certain factors, e. g. software type (infrastructure, UI, business logic etc.), project size, product development or customer-specific development, architecture constraints (from-scratch product development, product extensions, major refactoring, and renovation).

5 Introducing Lean Development

From industry customers who had long-standing experience with lean production and the Toyota production system (TPS) and from talks and publications by Mary and Tom Poppendieck [12,13,14] SAP management learnt about the lean development approach. This knowledge provided the business background for research into how lean development could be applied at SAP. It turned out that Scrum is particularily well suited to implementing lean software development:

1. Scrum's iterative cycles, called sprints, implement *takt*, i. e. the cyclic, repetitive work approach.
2. The *pull* principle of TPS (often discussed in conjunction with *Kanban*) can be found in Scrum in two places: (i) When planning a sprint the team "pulls" only so many requirements from the product backlog as can be fulfilled during the next sprint. Scrum does not allow for asking for more than the team promises. (ii) Each team member picks tasks from the sprint backlog when he or she sees fit. No manager assigns tasks to a team member.
3. The *Genchi Genbutsu* (go and see for yourself) principle is reflected by the product owner's participation in sprint reviews at regular intervals.
4. The *Kaizen* principle of continuous improvement is put into practice by regular retrospectives and the Scrum master's role to collect and communicate impediments and ideas for improvement to stakeholders.

In every major SAP development area (2000 to 5000 developers per area) a lean implementation team was set up to explore what had to be done to implement lean development processes. One team took over the pilot role by implementing their process model

six months earlier thereby gaining experiences to share with the other teams. These teams applied Scrum to their own change management projects.

It is worth to note that the pull principle had been in use at SAP for software configuration management for years. Change (delta) propagation from a development codeline to a test or consolidation codeline was usually done by a responsible person pulling the changes into the target codeline. This process was implemented for ABAP (SAP's proprietary application programming language) around 1990. For source code in other programming languages implementation took place in 2007.

5.1 Academic Experiments

Before applying Scrum on a large scale some experiments were carried out by the Hasso Plattner Institute, Potsdam (HPI). The HPI provides education for master and bachelor degrees in software systems engineering. As part of their practical education students have to carry out a half-year software development project with about 50 other students. These development projects were used to experiment with various approaches to upscaling Scrum, e. g. Scrum of Scrums, hierarchical backlog, hierarchy of product owners. Many insights were gained in these projects which proved beneficial to applying Scrum to multi-team projects at SAP [8]. Research on this topic is ongoing.

5.2 Continuous Improvement

In each development area a team was formed to collect and resolve problems the individual teams could not resolve themselves. These teams have to deal with both technical development problems and issues resulting from the new lean development approach. Therefore no limits are imposed on the types of problems communicated. These continuous improvement teams are also responsible for idea management in each area and serve as escalation contacts for Scrum masters.

5.3 Education

A one-day "lean awareness" training provided the background of the lean development approach. This training described the basic principles of lean production and development. To scale Scrum up fast to an entire organization Scrum mentors were trained by the existing Scrum mentors who had at least 1.5 years experience with Scrum in various contexts. This training was similar to the mentor trainings given earlier.

Scrum mentors held training courses for development teams, product owners, and Scrum masters. In order to provide training for all teams in a short time, the time allocated for training was reduced. Instead of training each team individually for one day as done during the pilot phase, two teams were trained together for half a day. Focus was put on in-depth training of Scrum masters and product owners.

People who attended the lean awareness training frequently commented that the knowledge about lean production helped them understand customer needs better, but the relationship between lean production and software development needed clarification. Many teams who had participated in the shortened training struggled later with communication and motivation issues. It turned out that in contrast to the pilot phase too

few Scrum mentors were available to deal with the numerous team problems. To overcome these problems team training sessions were extended to one day per team and the "lean mentor" role was created. Lean mentors underwent in-depth training in Scrum, lean methodologies, and communication. Scrum master and product owner training sessions were extended to two days each.

6 Scaling Scrum to the Max

Scaling agile software development processes is still subject to research [1,5,10,11]. Several approaches are known to co-ordinate the work of several Scrum teams working on one product, e.. Scrum of Scrums [9] or MetaScrum [18]. The size and complexity of SAP's products makes it necessary to consider these techniques. To co-ordinate the work of multiple Scrum teams and to communicate the requirements via team product backlogs, various approaches were combined. Special product teams were formed for this purpose. These teams can be regarded as both a permanent Scrum of Scrums and a supervisory product owner.

6.1 Product Teams

Product teams were introduced as a second organizational layer above the Scrum teams. A product team is responsible for the work of up to 7 development teams. It consists of the product owners of those teams and the same number of team members who are specialists in particular fields. Depending on the problem area other experts may be included. This allows for full engineering coverage of all problem areas expected, well-organized communication among the teams, and direct communication with the development teams to detect and mitigate risks. Typically the following expert roles can be found in product teams:

1. Chief product owner (responsible product manager),
2. Product team Scrum master (facilitator),
3. Software architect,
4. Delivery manager,
5. Knowledge management and product documentation expert,
6. User interface designer,
7. Stakeholder representative.

These roles may be taken by dedicated people or by development team members. For a product team to function properly, the following prerequisites have to be met:

1. Every member of the product team is also a member of one of the related Scrum teams.
2. The product team has full responsibility for requirements scope, quality, and delivery of the product. Each product team has a budget including external and travel costs.
3. No product team member is a line manager of another team member.
4. All product team members are collocated.

The tasks of the product team include: (a) product and budget management, (b) definition of the skill profiles, size, and numbers of Scrum teams, (c) assigning product backlog items to Scrum teams, (d) collecting project status information, (e) synchronizing the work of the Scrum teams, (f) working closely with the continuous improvement teams, (g) requesting supportive actions in case of impediments.

Three focus questions were dealt with in particular by dedicated roles: The chief product owner who is also the product owner of the product team has to answer the question *what* has to be delivered. The delivery manager decides *when* a particular functionality is to be delivered. The software architects decide *how* this functionality is to be implemented.

6.2 Area Product Teams

Product teams can rarely support more than seven Scrum teams. If the products or components are be too big or too complex to be developed in this organizational setup, an intermediate organizational layer is inserted between the product team and the Scrum teams. These teams are called "area product teams". Their responsibility is the same as the product teams' responsibility with the exception of budget and final product decisions. In this three-layer organization a product team works similar to a project management office. The complete budget and product responsibility is with the product team, while the operational day-to-day decisions are handled by the area product teams.

6.3 Observations

The Scrum roll-out happened very fast and was based on the assumption that during the pilot phase enough knowledge had been collected to go for a company-wide adoption. Upscaling methods were mainly driven by theoretical considerations. Implementing them revealed that the skills and knowledge of product managers and software architects were often not sufficient to manage a number of Scrum teams concurrently. Product teams had to communicate with business management, marketing, and field services who were not fully prepared for how Scrum and the product teams' expectations would change their daily work. The process of software requirements engineering were found to require a major adjustment.

Teams typically needed a full year to completely adopt Scrum. Later we could see signs of erosion especially in teams which had no Scrum mentor to support them. Team members often reported that they felt expected to work harder than was sustainable. Programmers complained about the pressure both from management and from within their teams. The first explanation given by the Scrum mentors was that these teams would pick more from the Scrum backlog than what they could implement. To counteract this they educated the teams about effort estimation and work-life balance. A detailed analysis revealed that the real root cause for demotivation and exhaustion was competition among the team members. They suspected that Scrum had been introduced to strive for higher efficiency and exchangeability of programmers, and a flatter management hierarchy with fewer career oportunities. As a result many programmers worked very hard to be visible and distinguishable from the others in the team.

Teams who had adopted Scrum to their satisfaction started looking into other agile methods. As of this writing many teams have adopted test-driven development [2] and

pair-programming [3]. This has been supported by trainings aiming at combining both methods into a team-wide, sustainable practice.

Cutural differences were not only important factors for Scrum adoption in the teams but also in management. We observed that Scrum was very differently introduced and implemented in various global locations, e. g. in Germany, the USA, Canada, France, India, Israel, and China. Work and management culture sometimes distorted Scrum to a means for additional career oportunities, more control, more documentation, more beaurocracy, or pressure for higher development speed, in stark contrast to the agile manifesto [4].

7 Conclusions and Outlook

The Scrum project management methodology has gained broad acceptance at SAP. Most teams consider Scrum their method of choice despite the many serious problems encountered. Employees who know SAP from its early days call it a dej vu although direct customer involvement in current development projects is very limited. Scrum has also been used for internal IT projects, change management, and marketing projects which all have in common that the exact goals could not be defined in the first place.

Scaling Scrum to a multi-team development organization is not as easy. Scrum of Scrums and backlog preparation for many teams can be combined by product teams as outlined above. Looking at all known parameters of agile project management at SAP we tend to say that not much more than 130 development employees may be organized in that fashion. This number sums up developers in 7 teams (max. 70 people), the product team (max. 16), development infrastructure responsibles (about 10), quality assurance and testers (about 25), general management (about 10).

A three-level hierarchy of teams to manage requirements for product backlog generation has not yet proven to work as expected. Currently the high complexity of SAP's products which is reflected in the organizational structure of SAP's development areas makes it impossible to drop one hierarchy level of project management. This observation has triggered significant effort to rigorously break up SAP's product lines into separate components to ensure that no product component needs more resources than the 130 people as calculated above.

To learn about the effects of lean development as implemented at SAP, the company chose to participate in the DIWA-IT study about health protection in the IT industry sponsored by the German Bundesministerium fr Bildung und Forschung and the European Union. An ongoing research project has been set up to examine long-term effects.

We understand lean development merely as a set of values and objectives rather than a framework of processes and best practices. There is still a lot of misunderstanding at SAP and in the world about lean development. Lean *production* processes have a long-standing tradition in industry, while lean *product development* processes have only recently started to be dicussed. It is dangerous to refer to lean production as a blueprint for product development. Both processes of value creation are completely different, and using terms from one process in the context of the other has often led to misinterpretations and confusion. We found that it is not enough to train software developers about Scrum and lean values. Significant effort needs to be put into educating people in

business administration and general management about innovation management and the pecularities of product design and development. In this respect software development is a very special case as the scientific management methods of Taylorism are unsuitable for processes with artifacts mostly existing in the minds of programmers.

We expect that after each product release the maintenance effort will increase for a couple of years when a new product gets widely adopted by customers. As discussed in a recent case study this maintenance effort requires additional considerations when creating the product backlog [20]. We consider the "competition" between new-product development and old-product maintenance a challenge to the long-term acceptance of Scrum at SAP. To counterbalance this effect organizational measures might be advisable.

Introducing lean development is a learning process which brings many problem areas to light. Many observations made at SAP would not have been made in the old environment. It is still too early to assess the benefits of the ongoing change in completeness, but the transparency of the development processes and the broad acceptance of agile methods gained make the change a worthwile effort. In addition to Scrum SAP is in the process of applying more and more of the developer-centered methods of Extreme Programming, e. g. test-driven development, pair programming, project metaphor, early code integration.

Looking back at the waterfall model that SAP used for about 10 years, the authors found that the biggest disadvantage of that model is not that it makes it impossible to correct errors at a later stage, but the strong impact the model has on organization and communication. The waterfall model suppresses communication along its path. Experts for one phase only communicate with experts for other phases through highly formalized documents and status data. Informal communication is regarded as undermining the model's simplicity. Unfortunately this attitude destroys the only remaining risk mitigation strategy left: communication.

Acknowledgements. The change described in this paper could not be carried out without a solid fundament of determined colleagues willing to accept delay and hardship while moving forward. In particular we wish to name Martin Fassunge, Alexander Gerber, Bernhard Gröne, Christiane Kuntz-Mayr, Christian Schmidkonz, and Jürgen Staader whose contributions were outstanding. We also would like to thank Jürgen Müller and Thomas Kowark of the Hasso Plattner Institute for their support to run the Scrum scaling experiments.

References

1. Ambler, S.W.: Scaling agile software development through lean governance. In: SDG 2009: Proceedings of the 2009 ICSE Workshop on Software Development Governance, pp. 1–2. IEEE Computer Society, Washington, DC, USA (2009)
2. Beck, K.: Test-driven development: By example. Addison-Wesley Professional, Boston (2002)
3. Beck, K.: Extreme programming explained: Embrace change, 2nd edn. Addison-Wesley Professional, Boston (2004)

4. Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R.C., Mellor, S., Schwaber, K., Sutherland, J., Thomas, D.: The agile manifesto (2001), http://agilemanifesto.org/
5. Eckstein, J., Josuttis, N.: Agile Softwareentwicklung im Großen: Ein Eintauchen in die Untiefen erfolgreicher Projekte. dpunkt, Heidelberg (2004)
6. Keller, F., Wendt, S.: FMC: An approach towards architecture-centric system development. In: ECBS, pp. 173–182. IEEE Computer Society, Los Alamitos (2003)
7. Knöpfel, A., Gröne, B., Tabeling, P.: Fundamental Modeling Concepts: Effective Communication of IT Systems. John Wiley & Sons, Chichester (2006)
8. Kowark, T., Müller, J., Müller, S., Zeier, A.: An educational testbed for the computational analysis of collaboration in early stages of software development processes. Accepted for HICSS 2011 (2011)
9. Larman, C., Vodde, B.: Scaling lean & agile development: Thinking and organizational tools for large-scale Scrum. Addison-Wesley, Upper Saddle River (2009)
10. Lee, G., Xia, W.: Towards Agile: An integrated analysis of quantitative and qualitative field data. MIS Quarterly 34(1), 87–114 (2010)
11. Leffingwell, D.: Scaling Software Agility: Best Practices for Large Enterprises. The Agile Software Development Series. Addison-Wesley Professional, Boston (2007)
12. Poppendieck, M., Poppendieck, T.: Introduction to lean software development. In: Baumeister, H., Marchesi, M., Holcombe, M. (eds.) XP 2005. LNCS, vol. 3556, p. 280. Springer, Heidelberg (2005)
13. Poppendieck, M., Poppendieck, T.: Implementing lean software development: From concept to cash. The Addison-Wesley signature series. Addison-Wesley Professional, Boston (2006)
14. Poppendieck, M., Poppendieck, T.: Leading lean software development: Results are not the point. The Addison-Wesley signature series. Addison-Wesley Professional, Boston (2009)
15. Schnitter, J., Mackert, O.: Introducing agile software development at SAP AG — Change procedures and observations in a global software company. In: 5th International Conference on Evaluation of Novel Approaches to Software Engineering, Athens, Greece, July 22-24, pp. 132–138 (2010)
16. Schwaber, K.: Agile project management with Scrum. Microsoft Press, Redmond (2004)
17. Schwaber, K.: The enterprise and Scrum. Microsoft Press, Redmond (2007)
18. Sutherland, J.: Future of scrum: Parallel pipelining of sprints in complex projects. In: ADC 2005: Proceedings of the Agile Development Conference, pp. 90–102. IEEE Computer Society, Washington, DC, USA (2005)
19. Takeuchi, H., Nonaka, I.: The new new product development game. Harvard Business Review 64, 137–146 (1986)
20. Vlaanderen, K., Brinkkemper, S., Jansen, S., Jaspers, E.: The agile requirements refinery: Applying Scrum principles to software product management. In: 3rd International Workshop on Software Product Management, Atlanta, Georgia, USA (September 1, 2009)