

Agile Methods for Large Organizations – Building Communities of Practice

Tuomo Kähkönen

Nokia Corporation

P.O. Box 407, FIN-00045 Nokia Group, Finland

E-mail: Tuomo.Kahkonen@nokia.com

Abstract

Agile development practices respect tacit knowledge, make communication more effective, and thus foster the knowledge creation process. However the current agile methods, like XP, are focused on practices that individual teams or projects need, and the use of the methods in organizations consisting of multiple cooperating teams is difficult. The Community of Practice theory suggests that large agile organizations should have various overlapping, informal cross-team communities. This paper studies three agile methods developed at Nokia that use facilitated workshops to solve multi-team issues. The paper explains using Communities of Practices theory - why these methods work in multi-team settings. The results of this paper suggest that workshop practices that amass people from different parts of organizations to perform a specific well-defined task can be used effectively to solve issues that span over multiple teams and to build up Communities of Practice. This result suggests that the Community of Practice concept could provide a basis for adapting agile methods for the needs of large organizations.

1 Introduction

Telecommunication companies are operating on increasingly dynamic markets. The new business challenges reflect on their product development, which has to be able to respond faster to changing situations. Several agile software development methods are proposed as an alternative to the traditional heavy weight approaches.

Agile methods like XP offer interesting ideas on how to improve team productivity and ability to adapt to changing requirements. However, the current agile methods are originally designed for small teams [1], and the applicability of the agile software development methods to large organizations is often considered challenging [2-4].

There are several experience reports portraying how agile methods have been implemented in various types of environments [e.g. 5-11]. Common to most of these cases is that there has been only one team involved at a time [12], and there are only a few reports of successful implementation of agile methods in medium-sized organizations consisting of dozens of developers [2, 3, 13, 14].

To solve the large and agile challenge, many authors have suggested that organizations should develop methods that combine agile and traditional elements [15-22]. Adaptive Software Development (ASD) [23] proposes a different approach: the building of collaborative practices, e.g. JAD (Joint Application Development) sessions, to bind teams together. Interestingly, this approach has received the least attention in empirical agile methods literature [24].

Organizational research has studied communication and coordination in large organizations for several decades and similar approaches can be found there. For organizations working in dynamic markets, the conditions are similar to those where agile software development methods are intended. In those kinds of environments, self-organizing communities that transcend official organizational structures have proven to be very centric enablers for success. Those communities are called Communities of Practice (CoP) in organizational research literature (e.g. [25, 26]).

Within Nokia, several in-house software development methods that can be considered as agile have been developed. Characteristic to these methods is that they are not directly based on XP and they have been successfully deployed in multi-team organizations. This paper introduces three of those methods and uses the CoP theory to analyze how they have solved the problems associated with multi-team communication and coordination.

Important similarities between the methods are identified. An important finding of this paper is that all methods studied use cross-team workshops in various

process areas to solve issues promptly, simultaneously creating working networks of people.

This paper is organized as follows: the next section introduces the key theoretical concepts used in this paper. Section three introduces the agile in-house methods studied and analyzes how they are promoting the creation of communities of practice. Section four discusses the findings, and the paper closes with final conclusions and implications for further research.

2 Related Research

2.1 Teams

Teams are a very widely used and studied management concept in general and also in software development. A team can be defined as a group of people with a common goal, interdependent work, and joint accountability for results [27]. Because team members share common goals and work in close proximity over an extended period, they easily develop a common identity and trust in each other. They easily share information and build on each other's ideas to solve business and technical problems. [27]

Although teams are widely used and solve many problems, they can also create some new problems. According to McDermott, teams are great tools for solving problems with functional silos in the organization, but at the same time they may create other type of silos. Teams can become insulated from each other. They reinvent tools, analyses, or approaches developed their peers on other teams. They waste time searching for information they know one of their colleagues has. When teams have little contact with other teams, they can get into habit of rejecting outside ideas and lose their ability to generate new ideas. [27]

Agile methods are team-focused solutions to software development. E.g. in XP, a team is a closed unit consisting of a customer and the development team. All input to the team comes through the customer; there are no practices on how to combine two teams using XP.

Adaptive Software Development [23] is an exception and it discusses the multi-team issues to some extent. Highsmith emphasizes that teams must collaborate and improve their joint decision-making ability when building more adaptable lifecycles. [28]

2.2 Communities of Practice

An important concept in organizational research, overcoming limitations imposed by teams, has been the

Community of Practice (CoP). It is defined as people who are bound by informal relationships and share a common practice. [25, 26, 29] *Community* refers to the informality and personal basis of many relationships in typical CoPs. *Practice* indicates that CoPs are centered on shared practices. CoP boundaries do not correspond to official organizational boundaries but rather to practice- and person-based networks. People in the community are contextually bound by shared interest in learning and applying a common practice. This results in an emergent nature for the CoPs [30].

Communities of Practice are ubiquitous, everybody belongs to several communities. They have existed as organizations even before the introduction of the concept was acknowledged, albeit unofficially. Previously organizational research has focused on formal structures, like teams, and the official communication that happens between them, like documents. CoPs transcend traditional organizational boundaries and the theory behind them also acknowledges the importance of tacit knowledge and the social aspects of knowledge creation.

From the point of view of a large software development organization, this concept is interesting, because in large software development organizations there are multiple dimensions and the formal organizational structure typically reflects only one or two of those dimensions [31]. CoPs can thus cover the dimensions that the formal organizational structures do not cover.

The theory of CoP suggests that in addition to the team structure there should be multiple overlapping communities that transcend team boundaries for sharing knowledge and standardizing practices. [32] Thus in agile methods there should be also practices that put people in different teams together similar to the way that XP puts people together within teams.

Although CoPs arise naturally, organizations can influence their development [32-35]. CoP researchers have provided various lists of the kinds of activities organizations should do in order to support and cultivate CoPs [27, 32, 34-36]. E.g. McDermott [32] gives the following guidelines for building Communities of Practice:

- Build communities around a few important topics
- Find and build on natural networks
- Develop community coordinators and core groups
- Initiate some simple knowledge-sharing activities
- Support communities (e.g. time and encouragement)
- Create a community support team
- Be patient

How to build CoPs in software development organizations in particular has not yet received the attention of researchers. However, many ideas presented in Adaptive Software Development (ASD) [23] are aligned with the ideas represented by the CoP theory. This offers some basis for seeking the needed practices, because ASD

offers JAD (Joint Application Development) sessions and other similar practices as a solution for enhancing cross-team collaboration in agile multi-team software development organizations.

A JAD session is basically a facilitated workshop for software requirements definition. Facilitated workshops as such are nothing new, they were developed originally for use outside the software engineering domain, and there are now books about workshop facilitation in general (e.g. [37, 38]).

Facilitated workshops have also been used in software development since the late 1970s. Influenced by the group dynamics discipline and the study of group work and meetings [39], Morris and Crawford developed the Joint Application Development (JAD) method at IBM [40]. Since then, many companies have taken it into use and there have been many variants of the method under different brand names [39, 41]. However, there is one thing they all have in common – the facilitated sessions [41].

Highsmith [42] proposes that this kind of practice should be employed when the agile methods are scaled up for larger organizations. He has also proposed a new definition for JAD sessions: “a facilitated workshop that brings together cross-functional groups to build collaborative relationships capable of producing high quality deliverables during the life of the project” [23, p.135]. It is worth noting that this definition extends the scope of the JAD sessions beyond its original scope of the requirements definition.

3 Agile Methods Using Facilitated Workshops

This chapter introduces how three agile methods developed at Nokia have used facilitated workshops to promote cross-team communication.

The methods were developed independently in different Nokia business units and they were discovered while surveying the use of agile software development practices at Nokia [43]. Several units using agile development practices to varying degrees were identified, and the methods studied in this paper were selected because they fulfill the following three criteria: they were solving multi-team management issues using collaborative practices, they were fulfilling the definition of agile development, and the methods were institutionalized in the organizations.

There are plenty of definitions of agile software development [43-46]. Using the Abrahamsson et al [44] definition of agile development, the selected methods can be considered as agile because they are incremental, cooperative, straightforward and adaptive.

All the studied methods have been used successfully in multi-team software development organizations. The use of the methods can be considered as successful because the methods have been institutionalized in the organizations [47] and, based on interviews, there has been positive feedback from people using the methods and from management. The approach to solving the multi-team management issues in all of the cases has been the introduction of collaborative cross-team workshop practices similar to those proposed by Highsmith in the ASD method [23].

3.1 RaPiD7

The RaPiD7 [48] method defines a rapid way to elaborate documents. In traditional document-driven SW development the documents have become a goal on their own, and this over emphasis of documentation is often criticized [49]. In the traditional approach one person takes a relatively long time to write a document, and, when he thinks it is ready, the stakeholders review it. The feedback from the stakeholders comes in the final stages of elaboration of the document.

RaPiD7 addresses this problem by involving the relevant stakeholders already in the early stages of document writing. This is achieved in practice by organizing workshops where the documents are written. After the workshop, typically only a small amount of editorial work is needed to finalize the artifact.

Compared with the traditional approach, more people are participating in the document creation and the effort put in is about the same, but the calendar time needed to produce the documents is reduced. This effect is especially important when the project follows a lifecycle where documents are produced sequentially. The goal is to have an approved document version one week after the original workshop.

As the name of the method suggests, there are 7 steps in the workshop process (see figure 1). The purpose of preparation is to be prepared enough for an efficient workshop that has a defined goal and the necessary people with the right information. The workshop is started with a kick-off to gain a common understanding of the goal, scope and terminology of the specific workshop. Then the participants start to find and create possible ideas for a specific part of the artifact to be produced by brainstorming, discussions and by going through preliminary material. In the analysis the initial set of ideas that will be further developed and specified in the workshop is selected, and in the detailed design those ideas are developed further and documented. In the decision step the most feasible solutions are selected and

the consistency between different solutions is checked. Finally, in the closing it is decided whether the desired goals have been met or should there be additional workshops.



Figure 1: RaPiD7 process [48]

RaPiD7 workshops are not organized as isolated events but a master plan about which workshops are needed throughout the project does exist. Each workshop has a facilitator who coordinates the practical arrangements and conducts the workshop. There are also training programs for workshop participants and new facilitators.

The participants of each the workshop depend on the scope of the workshop. If there is thought to be dependencies that were not fully covered in the workshop due to missing stakeholders, the document goes through the normal review process.

RaPiD7 is mainly used to create different kinds of specification documents, ranging from the product to module level. Typical documents produced are functional and technical specifications, test plans, project plans and customer documents. Also test drivers and process documents have been produced using this method. RaPiD7 does not define which artifacts are produced. It depends on the project and the process model variant it is using.

RaPiD7 is used in several units at Nokia. It was originally developed in a software development program consisting of multiple projects, but it has been applied also in software product line types of organizations. The organization has established process models that the projects are expected to follow. The process model defines the project milestone framework and the artifacts to be produced.

3.2 Integration Camp

The case organization is a large SW product line consisting of about 20 teams. One of the teams is an integration team responsible for official releases every second week. The development teams release their components and associated tests to the integration team, which integrates them with the external components. After successful regression and system tests, the integrated release is delivered back to the SW development teams and the system platform program.

As the SW product line is not very old, new components are introduced frequently. Smooth introduction of the new components into the common integration process is a crux of the process because the integration process is the bottleneck that sets the pace for the organization’s release cycles.

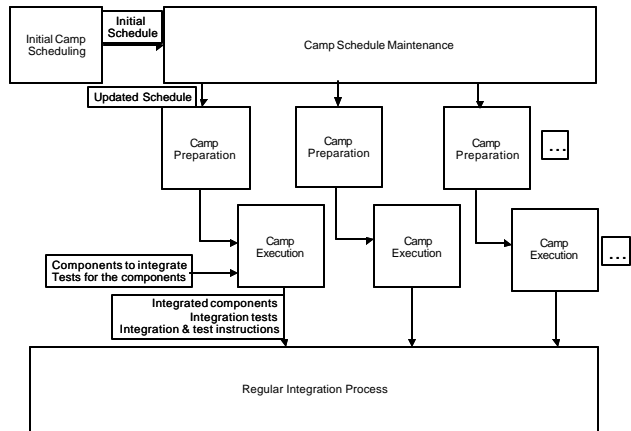


Figure 2: Integration Camp process

Traditionally in similar organizational settings the introduction has happened when the component was “ready” by sending an e-mail to the integration team to inform them of the component’s availability for integration at the configuration management system. The first integration has typically been tedious because the integration team has not been aware of how the component works. After many phone calls and a couple of formal change requests the component was finally successfully integrated, but in the next release the same or very similar problems typically occurred. In this mode it took several integration cycles before the integration of the new component ran smoothly and with minimal effort.

With many new components coming in simultaneously and relatively short release cycles, the old approach was seen as inadequate. To solve the problem, the integration team came up with the Integration Camp (figure 2) idea.

The idea behind the Integration Camp is that the development team and the integration team are working together for the first integration. As soon as the component has some functionality that can be released and the code is so stable that it can be included in the official build, the development team signs up for an Integration Camp.

To participate in the camp, the component development team must fulfill certain prerequisites. It must provide e.g. architecture documents and test cases for the components. Each camp has clearly defined goals about what it is supposed to achieve – typically the goal is to integrate and test the component.

There are typically 2-10 participants in each camp and the camp takes 1 day (although sometimes up to 5 days). The participants depend on the component and the goals to be achieved. In addition to the integration team and the component developers, there can be representatives from other factories or customer programs. Integration Camps have a facilitator who schedules the camps with the participants, takes care of the practical arrangements, conducts the camp and compiles the minutes.

The camp is held in one meeting room equipped with the necessary hardware. The camp starts with a kick-off meeting, where the camp goals and some important technical issues are discussed. Participants then agree on the tasks that everyone will take on. The components are integrated in parallel, and, if problems arise, they are solved together.

Once the component is successfully integrated, a testing session follows. The testing session is important for succeeding also in the next builds. As the camp produces a complete integration test set, it is possible to verify the next versions of the components. The camp is finalized with a wrap-up meeting where the camp activities are reviewed and the remaining action points and recurring actions for subsequent releases are agreed upon.

3.3 SEED

The SEED method [50] is an evolutionary software development model utilizing some ideas from Gilb's [51] evolutionary project management. It covers the whole software engineering process, from requirements to integration at the sub-system level and also corresponding project management activities (see figure 3).

The SEED method has been developed for several years based on the organization's needs. The organization has also a traditional process model and milestone framework, but it was noted that those modeling techniques are not able to capture all relevant aspects of

the work carried out in the organization. For that reason, a new type of process documentation technique was needed to complement the traditional techniques.

The pattern approach (see e.g. [52, 53]) was found to be a suitable solution, and the actual practices were documented retrospectively using that technique. In the SEED method, patterns are called behavior patterns to emphasize that they cover a wider range of issues than processes issues only. The behavior patterns are divided according to Leavitt's diamond [54] into structure, task, actor and technology patterns.

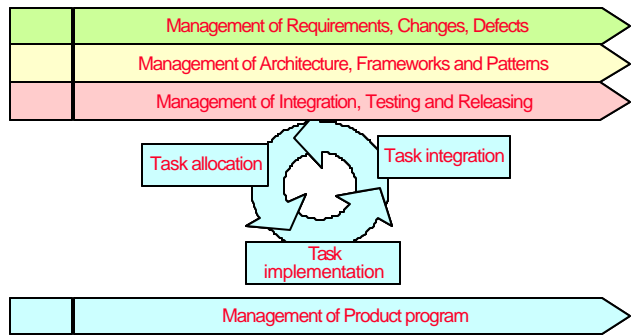


Figure 3: SEED process [50]

The SEED method has a traditional milestone model that defines project phases, key activities in them and the milestones between the phases. This milestone model complies with the organizational requirements both with respect to the milestone requirements and the presentation. This layer gives a kind of mapping between organizational process and quality requirements and the elements in the SEED process and thus helps to get formal acceptance of the agile approach. As the organization is developing a subsystem, this framework also helps to integrate the organization's processes to the system level processes.

In SEED high-level system requirements are managed using traditional requirements management process. The organization developing the subsystem using the SEED process takes those requirements as given. The high-level requirements are stored in a requirements management tool, but the detailed ones are in a requirements specification document.

To manage requirements and architecture, SEED has an established concept group that has a representative from all teams. It analyzes the high-level requirements and identifies possible dependencies between components and teams. The concept group makes decisions about the internal architecture and decides to which development team each feature is allocated for detailed design and

implementation. System-level architectural issues are escalated to the corresponding system-level board.

Project planning and tracking is done in a weekly project manager meeting that has a project manager from each team. Project managers prepare for the meeting by updating the current situation and plan of the team. The plans are synchronized in the weekly project management meeting and, at the same time, the status of each team is communicated to other project managers.

SEED defines a complete project planning and tracking model. Tasks are used to allocate the work within the team. There are four kinds of tasks: feature development, re-engineering, defect correction and open issue resolution. Tasks are small, up to 2 weeks at most. They are also goal-oriented, defining end results of the work rather than activities to be performed. Re-engineering and open issue tasks are also time-boxed so that they don't take more than 2 weeks at a time. The tracking is done based solely on the functionality completed, not the tasks. As the tasks are supposed to be result-oriented, there is no mismatch.

The key planning and tracking tool in SEED is the Coming Feature List document that indicates the content of forthcoming releases. The document is updated in the weekly project manager meeting. Due to the high number of changes, all changes are documented and reviewed in the meeting.

4 Discussion

4.1 Communities

Various communities that use facilitated workshops as a key practice for managing multi-team issues can be identified from the case organizations (see Table 1). Communities are built around key process areas where coordination is needed. Each case has selected one or two different process areas to start with. Communities have been grown around a core group of people belonging to the official organizational structure. The core group is a permanent team in the case of Integration Camp, whereas the RaPiD7 community is built around a project team, and in SEED around a workgroup of specialists of a certain discipline. Although there initially may seem to be many approaches, these observations conform to McDermott's suggestions to build communities around a few important topics and natural networks [32].

The examples showed successful implementation of facilitated workshops in architecture management, design project management and integration. The diversity in the field of application of facilitated workshops leads to another important observation: the facilitated workshops

can be used practically in any software process area, not only in requirements development and design as suggested by many JAD types of approaches. This observation aligns with Highsmith's extended definition of JAD workshops [23, p.135].

Table 1: Summary of CoPs in the case organizations

	Community	Practice
RaPiD7	People interested in a specific system. Includes development team, related teams and specialists	System specification is elaborated and documented in workshops
IC	People delivering software for a common platform	Software is integrated in workshops
SEED	Project managers of the subsystem	Subsystem development is planned and tracked in workshops
SEED	Concept Group = Team Architects	Requirements and architecture of the subsystem are managed in workshops

4.2 Workshop Practices to Build the Communities of Practice

Several commonalities in the workshop practices could be identified. This is interesting, since the methods were developed independently. Moreover, in all cases the practices developed have followed to a large extent the recommendations for building up communities of practice, although this has not been the explicit goal.

Continuity. Workshops are not isolated, but either the workshops are scheduled to take place at a certain time or a master plan exists for which workshops are needed and when. This creates a rhythm for the community, which is seen as an important characteristic of dynamic communities [34]. One manifestation of the continuity of the practices has been the fact that the practices have been retained even if the projects have changed. In RaPiD7 and Integration Camp, there is an explicit coordinator role to ensure the continuity of the practice.

Procedure. In all methods there is a clear procedure on how the workshops are organized, and the procedures are very similar. It may be argued that the workshop procedure is the practice around which the community is

built. The procedure states e.g. that the expectations for the workshop should be clearly set, the participants are selected according to the goals and are preparing for the workshop, and each workshop has a facilitator and a prior agenda.

Participants are selected based on the problem in question, not by formal organizational structures. The CoP theory uses the term ‘legitimate peripheral participation’ [25] to describe that the community is open to new people. The internal customer for the deliverable produced is often present in the workshop.

Documentation. In all three methods workshops are important tools to create new knowledge and also to document it in the artifacts. The artifacts are elaborated mainly in the workshops. The minutes of the workshop are written and action points are tracked.

Management support in the form of legitimacy, time and resources is important for the formation of CoPs. Workshops seem to be a readily accepted practice by management because the workshops are delivering concrete results and the benefits are visible soon. The social interactions that develop and maintain shared knowledge and personal networks seem to be by-products of the used practices, not explicit goals.

4.3 Implications for Agile Method Development

Research has shown that software development is highly knowledge-intensive work [e.g. 55, 56], and a centric role of tacit (i.e. undocumented) knowledge in knowledge creation in general [57-59] and in agile method in particular [19, 60, 61] has been highlighted. CoP is a concept that helps to leverage the tacit knowledge in a multi-team organization in a similar way that the current agile methods do in the scope of one team.

Current agile methods are largely missing the practices that bind two teams together. E.g. XP practices enable efficient communication between the team members, acknowledge the importance of tacit knowledge and in this way foster knowledge creation [60]. However, all this happens within the scope of the team. There are no explicit practices that link the team members to other teams in the organization.

The CoP theory thus suggests that when building agile methods for multiple cooperating teams, there should be practices that help build up informal communities that transcend traditional team boundaries.

This proposition is important for agile method developers, since it states that it is not sufficient if a multi-team organization uses agile methods within the team

scope and applies traditional communication practices between the teams. Instead, method developers should look for effective practices to enhance collaboration between teams in different kinds of communities.

JAD sessions have been suggested as one practice to manage requirements in agile multi-team organizations [23]. The results of this paper indicate that facilitated workshops can be used also in many other process areas besides requirements development. Examples of the use of the workshops in architecture management, design, software integration and project management were provided. This finding aligns with Highsmith’s extended definition of the JAD workshops [23].

5 Conclusions

Use of agile software development methods in large organizations has been challenging. This paper opened up new directions for scaling up the use of agile approaches by introducing CoPs as an explanatory theory. Using this theory, the paper pointed out one important limitation of most of the current public agile methods: they rely on a team concept that is insufficient for success in large organizations operating on dynamic markets. The results of this paper supported the proposition that in addition to the official team structures, there should be in place CoPs that transcend traditional team boundaries. For agile methods, this proposition implies that they should incorporate also practices that contribute to the creation and effective operation of such communities.

This paper studied in more detail one practice, facilitated cross-team workshops, which is beneficial for the formation of such communities. The paper analyzed how such workshops are used in three agile software development methods developed at Nokia. The analysis provided important insights that are valuable for organizations developing agile methods in similar settings.

It was found that it is possible to use cross-functional workshops in any process area, not just in requirements definition, like in many variants of JAD techniques. Examples of the use of the workshops in architecture management, design, software integration and project management were provided. This finding is aligned with Highsmith’s extended definition of the JAD workshops [23].

In each case it was possible to identify a community that was built around an official organizational unit. Facilitated cross-team workshops, while a simple and intuitive practice, were perceived to be an effective practice for cultivating knowledge in these communities. It was also found that individual workshops alone are not sufficient, but the workshops must form a continuum.

The techniques utilized in the workshops were found to be consistent with the general guidelines for organizing workshops, i.e. there must be a workshop facilitator, they must have a clearly defined scope, etc.

For practitioners, the results of this paper offer one concrete practice that enables agility in large organizations. It has been recommended that building up CoPs should be initiated with some simple knowledge-sharing activities [32]. Workshops are one such practice that can be started by using pure common sense. There are, however, studies on how to organize effective workshops, and the teams may benefit by getting familiar with that information.

For researchers, this paper points out new directions on how to proceed with the study of agile methods. By using the CoP theory, it is possible to explain why certain methods do work in multi-team settings while others have difficulties. Method developers in particular will benefit from the guidance the CoP theory offers for further development of the agile methods.

The focus of this paper has been on facilitated workshops. It should be noted that cross-functional workshops are not the only practice that helps in the building up of CoPs. Thus more research is needed to find other practices that can be used in large software development organizations to further support the creation and operation of CoPs. Generic research done with regards to CoPs will give valuable input for that research.

References

- [1] K. Beck, *Extreme Programming Explained: Embrace Change*. Upper Saddle River, NJ: Addison-Wesley, 2000.
- [2] M. Simons, "Big and Agile?," *Cutter IT Journal*, vol. 15, pp. 34-38, 2002.
- [3] S. Murthi, "Scaling Agile Methods - Can eXtreme programming work for large projects?," *New Architect*, 2002.
- [4] P. Allen, "Light Methodologies and CBD," *Component Development Strategies*, vol. XI, pp. 1-16, 2001.
- [5] J. Grenning, "Extreme Programming and Embedded Software Development," presented at Embedded Systems Conference 2002, Chicago, 2002.
- [6] P. Schuh, "Recovery, Redemption, and Extreme Programming," *IEEE Software*, vol. 18, pp. 34-41, 2001.
- [7] A. Anderson, R. Beattie, K. Beck, D. Bryant, M. DeArment, M. Fowler, M. Fronczak, R. Garzaniti, D. Gore, B. Hacker, C. Handrickson, R. Jeffries, D. Joppie, D. Kim, P. Kowalsky, D. Mueller, T. Murasky, R. Nutter, A. Pantea, and D. Thomas, "Chrysler Goes to "Extremes". Case Study.," *Distributed Computing*, pp. 24-28, 1998.
- [8] O. Murru, "Assessing XP at a European Internet Company," *IEEE Software*, pp. 37-43, 2003.
- [9] J. Rasmusson, "Introducing XP into Greenfield Projects: Lessons Learned," *IEEE Software*, pp. 21-28, 2003.
- [10] J. R. Nawrocki, B. Walter, and A. Wojciechowski, "Comparison of CMM level 2 and eXtreme programming," presented at 7th European Conference on Software Quality, Helsinki, Finland, 2002.
- [11] M. M. Müller and W. F. Tichy, "Case Study: Extreme Programming in a University Environment," presented at 23rd International Conference on Software Engineering, Toronto, 2001.
- [12] P. Abrahamsson, "Extreme Programming: First Results from a Controlled Case Study," presented at 29th Euromicro Conference, Belek, Antalya, Turkey, 2003.
- [13] J. Sutherland, "Agile Can Scale: Inventing and Reinventing SCRUM in Five Companies," *Cutter IT Journal*, vol. 14, pp. 5-11, 2001.
- [14] D. Karlström and P. Runeson, "Scaling Extreme Programming in a Market Driven Development Context," presented at XP2003, Genova, Italy, 2003.
- [15] M. Paulk, "Agile Methodologies and Process Discipline," *CrossTalk The Journal of Defense Software Engineering*, pp. 15-18, 2002.
- [16] M. Fowler, "Is Design Dead," in *Extreme Programming Examined*, G. Succi and M. Marchesi, Eds.: Addison-Wesley, 2001, pp. 3-18.
- [17] B. Henderson-Sellers, "Agile or Rigorous OO Methodologies: Getting the Best of Both Worlds," *Cutter IT Journal*, vol. 15, pp. 25-33, 2002.
- [18] B. Boehm, "Get Ready for Agile Methods, with Care," *IEEE Computer*, pp. 64-69, 2002.
- [19] B. Boehm and R. Turner, *Balancing Agility and Discipline*. Boston: Addison-Wesley, 2003.
- [20] R. Turner and A. Jain, "Agile Meets CMMI: Culture Clash or Common Cause?," presented at 2nd XP and 1st Agile Universe Conference, Chicago, IL, 2002.
- [21] J. Highsmith, "What Is Agile Software Development?," *CrossTalk The Journal of Defense Software Engineering*, pp. 4-9, 2002.
- [22] D. Reifer, "XP and the CMM," *IEEE Software*, pp. 14-15, 2003.
- [23] J. Highsmith, *Adaptive Software Development*. New York: Dossert House Publishing, 2000.
- [24] P. Abrahamsson, J. Warsta, M. T. Siponen, and J. Ronkainen, "New directions on agile methods: A comparative analysis," presented at International Conference on Software Engineering (ICSE25), Portland, Oregon, USA, 2003.

- [25] J. Lave and E. Wenger, *Situated Learning: Legitimate Peripheral Participation*. Cambridge: Cambridge University Press, 1991.
- [26] J. S. Brown and E. S. Gray, "Organizational learning and communities-of-practice: Towards a unified view of working, learning and innovation," *Organization Science*, vol. 2, pp. 40-57, 1991.
- [27] R. McDermott, "Learning Across Teams: The Role of Communities of Practice in Team Organization," *Knowledge Management Review*, vol. 2, 1999.
- [28] J. Highsmith, "Retiring Lifecycle Dinosaurs," *Software Testing & Quality Engineering*, pp. 22-28, 2000.
- [29] J. Lave and E. Wenger, "Situated Learning: Legitimate peripheral participation," Institute for Research on Learning, Palo Alto IRL Report 90-0013, 1990.
- [30] W. Snyder, "Communities of Practice: Combining Organizational Learning and Strategy Insights to Create a Bridge to the 21st Century," vol. 2003, 1997.
- [31] J. Bosch, "Software Product Lines: Organizational Alternatives," *IEEE Software*, 2001.
- [32] R. McDermott, "Nurturing Three Dimensional Communities of Practice: How to get the most out of human networks," *Knowledge Management Review*, vol. 2, 1999.
- [33] E. Wenger, "Communities of Practice: Learning as a Social System," *Systems Thinker*, vol. 9, 1998.
- [34] E. Wenger, R. McDermott, and W. Snyder, *Cultivating Communities of Practice: A Guide to Managing Knowledge*. Boston: Harvard Business School Press, 2002.
- [35] G. Lintern, F. J. Diedrich, and D. Serfaty, "Engineering the Community of Practice for Maintenance of Organizational Knowledge," presented at IEEE 7th Human Factors Meeting, Scottsdale Arizona, 2002.
- [36] P. Gongla and C. R. Rizzuto, "Evolving communities of practice: IBM Global Services experience," *IBM Systems Journal*, vol. 40, pp. 842-862, 2001.
- [37] S. Kaner, L. Lind, C. Toldi, S. Fisk, and D. Berger, *Facilitator's Guide to Participatory Decision-Making*. Philadelphia: New Society Publishers, 1996.
- [38] T. Justice and D. Jamieson, *The Facilitator's Fieldbook*. New York: AMACOM, 1999.
- [39] E. Carmel, R. D. Whitaker, and J. F. George, "PD and Joint Application Design: A Transatlantic Comparison," *Communications of the ACM*, vol. 36, pp. 40-48, 1993.
- [40] A. Crawford, *Advancing Business Concepts in a Jad Workshop Setting: Business Reengineering and Process Redesign*: Prentice Hall, 1994.
- [41] M. Yatco, "Joint Application Design/Development," vol. 2003, 1999.
- [42] J. Highsmith, *Agile Software Development Ecosystems*. Boston, MA.: Pearson Education, 2002.
- [43] J. Vanhanen, J. Jartti, and T. Kahkonen, "Practical Experiences of Agility in the Telecom Industry," presented at XP 2003, 2003.
- [44] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, *Agile software development methods: Review and Analysis*. Espoo, Finland: Technical Research Centre of Finland, VTT Publications 478, online: <http://www.inf.vtt.fi/pdf/publications/2002/P478.pdf>, 2002.
- [45] P. Kruchten, "Agility with RUP," *Cutter IT Journal*, vol. 14, pp. 27-33, 2001.
- [46] G. G. Miller, "The Characteristics of Agile Software Process," presented at The 39th International Conference of Object-Oriented Languages and Systems (TOOLS 39), Santa Barbara, CA, 2001.
- [47] S. Zahran, *Software Process Improvement Practical Guidelines for Business Success*. Harlow: Addison-Wesley, 1997.
- [48] R. Kylmäkoski, "Efficient Authoring of Software Documentation Using RaPiD7," presented at 25th International Conference on Software Engineering ICSE 2003, Portland, Oregon, 2003.
- [49] R. Glass, "Agile Versus Traditional: Make Love, Not War!," *Cutter IT Journal*, vol. 14, pp. 12-18, 2001.
- [50] J. Rikkilä, "Agile Process in the Medium Size Organization - a Product Development View," 2002.
- [51] T. Gilb, *Principles of Software Engineering Management*. Wokingham: Addison-Wesley, 1988.
- [52] C. Alexander, *The Timeless Way of Building*. New York: Oxford University Press, 1979.
- [53] J. Coplien, *Software Patterns*. New York: SIGS Books & Multimedia, 1996.
- [54] H. J. Leavitt, "Applied organizational change in industry: Structural, technological, and humanistic approaches," in *Handbook of Organizations*, J. G. March, Ed. Chicago: Rand McNally, 1965, pp. 1144-1170.
- [55] L. Prusak, *Knowledge in organizations*. Oxford: Butterworth-Heinemann, 1997.
- [56] M. Robertson, C. Sørensen, and J. Swan, "Survival of the leanest: Intensive knowledge work and groupware adaptation," *Information Technology & People*, vol. 14, pp. 334-352, 2001.
- [57] M. Polyani, *The Tacit Dimension*. New York: Anchor, 1967.
- [58] I. Nonaka and H. Takeuchi, *The Knowledge Creating Company: How Japanese Companies Create the Dynamics of Innovation*. New York: Oxford University Press, Inc., 1995.

[59] I. Tuomi, *Corporate Knowledge: Theory and Practice of Intelligent Organizations*. Helsinki: Metaxis, 1999.

[60] T. Kähkönen and P. Abrahamsson, "Digging into the Fundamentals of Extreme Programming - Building the Theoretical Base for Agile Methods," presented at Euromicro 2003, Belek, Antalya, Turkey, 2003.

[61] A. Cockburn, "Agile Software Development Joins the "Would-Be" Crowd," *Cutter IT Journal*, vol. 15, pp. 6-12, 2002.