Understanding Decision-Making in Agile Software Development: a Case-study

Nils Brede Moe, Aybüke Aurum
SINTEF ICT, Norway
School of Information Systems, Technology and Management, University of New South Wales, Sydney, Australia
nils.b.moe@sintef.no, aybuke@unsw.edu.au

Abstract

A challenge with introducing agile software development is changing the way decisions are made. In this paper, we discuss the decision making processes used in Scrum teams. We found that a prerequisite for introducing Scrum is the alignment of decisions on a strategic, tactical and operational level. In addition, specialisation can be a barrier for the decision-making process on the operational level, and that daily meetings are important for preventing decision-hijacking. Also removing the hold-up problem makes it easier for developers to participate in the decision-making process.

1. Introduction

Agile software development represents a new approach for planning and managing software projects [11]. Agile development differs from the traditional plan-driven approaches as it puts less emphasis on up-front plans and strict plan-based control and more emphasis on mechanisms for change management during the project [16]. Further, agile development relies on people and their creativity rather than on processes [8]. Leadership and collaboration, informal communication and an organic (flexible and participative, encouraging cooperative social action) organizational form are other characteristics of agile software development [17].

Cohn and Ford [10] have referred to a number of possible setbacks or errors that may occur when an organisation is making the transition from plan-driven towards agile processes. The problems are mainly caused by resistance to, or over-enthusiasm for, agile practices within a software development team. When changing from plan-driven to change-driven (agile), this may also impact several aspects of the organization including its structure, culture, and management practice [17]. Neither culture nor mind-sets of people can be easily changed, which makes the move to agile methodologies all the more formidable for many organizations [7].

One of the reasons for why agile development has attracted much interest is because of claims of many improvements on areas such as work performance, quality and work environment. However there is a lack of scientific support for many of these claims [11]. Especially management-oriented approaches are clearly the most under-researched area compared to it's popularity in industry [11].

Software development processes depend significantly on team performance, as does any process that involves human interaction. Agile development relies on teamwork, as opposed to individual role assignment that characterizes plan-driven development [17]. A team following a plan-driven model often consists of independently focused self-managing professionals, while an agile team is supposed to be fully self-organized [16]. Tata and Prasad [22] claim that self-organized teams directly influence team effectiveness since it brings decision-making authority to the level of operational problems and, thus, increase the speed and accuracy of problem solving. Also, such teams are able to make decisions with economic consequences [14]. Introducing self-organizing teams and change the way project decisions are made, is probably one of the biggest changes when introducing agile software development [23]. To the best of our knowledge there are no studies that describe how this can be done in practice.

This article presents the results of an empirical study, based on data collected from two software projects recently adapting Agile methods. The main objective of this paper is to provide insight into the decision making processes used in agile software teams. Our research question is:

How does decision-making theory explain the challenges involved in introducing Agile development?

978-0-7695-3276-9/08 $25.00 © 2008 IEEE
DOI 10.1109/SEAA.2008.55

IEEE computer society

The remainder of the paper is organized as follows. In Section 2, we describe agile teams, self-organizing teams and a model for decision-making. In section 3, we describe our research method in detail. In Section 4, we present results from a multiple-case study on the decision-making process in agile software teams. We discuss our findings in Section 5. Section 6 concludes and presents suggestions for future research

2. Background

In this section we present background information on agile development, and we use the literature to describe how self-organizing teams and decision-making is related, and why this is important in agile development.

2.1. Agile Development and Scrum

Agile software development comprises a number of practices and methods [1, 9]. Among the most known and adopted agile methods are Extreme Programming (XP) [6] and Scrum [20]. XP focuses primarily on the implementation of software, while Scrum focuses on agile project management [2]. In this study the focus is on Scrum as Scrum encourages self-organizing teams and focuses on project management.

The name "Scrum" was inspired by an article characterizing new product development teams in Japan, by Takeuchi and Nonaka [21], where "self-organizing project teams" was one of key characteristics. The Scrum-team is given significant authority and responsibility for many aspects of their work, such as planning, scheduling, assigning tasks to members, and making decisions [20]. Scrum focuses on project management in situations where it is difficult to plan ahead, with mechanisms for "empirical process control"; where feedback loops is the core element [20].

Scrum and Agile development favour a leadership-and-collaboration style of management where the traditional project manager's role is replaced with the Scrum-master's role of a facilitator or coordinator [1, 9]. The Scrum-master is in charge of solving problems that stops the Scrum-team (5-9 people) from working effectively. The Scrum-master is often described as a coach or facilitator. He or she does not organize the team (designers and developers); the team organizes itself and makes the decisions concerning what to do. The Scrum-master works to remove the impediments of the process, runs and makes decisions in the daily meetings and validates them with the management [20]. Hence, Scrum-master need to work with the product owner side by side.

Software is developed by the self-organizing team in increments (called "sprints"), starting with planning and ending with a review. The team coordinates on a daily basis. Features to be implemented are registered in a "backlog", and a product-owner decides which backlog items should be developed in the following sprint. These items are specified in a sprint backlog.

The product backlog defines everything that is needed in the final product based on current knowledge. The backlog comprises a prioritized and constantly updated list of business and technical requirements for the system being built or enhanced. Backlog items can include features, functions, bug fixes, requested enhancements and technology updates. Multiple stakeholders can participate in generating product backlog items, such as customer, project team, marketing and sales, management and support [1]. Prioritizing the backlog is a complex communication and negotiation process; however the product-owner is the one responsible for the final prioritizing. During the planning-meeting (usually every second or forth week) the product-owner is responsible for presenting a prioritized product backlog.

While the product backlog is constantly updated, the sprint backlog should not be changed during the sprint, unless critical business premises suddenly change or if the team somehow is not able to deliver as planned.

2.2. The Self-organizing Team

We use the label "self-organizing" teams as a synonym for "autonomous teams" and for "empowered teams". Guzzo and Dickson [14] describe such teams as a group of employees who typically perform highly related or interdependent jobs, who are identified and identifiable as a social unit in an organization, and who are given significant authority and responsibility for many aspects of their work, such as planning, scheduling, assigning tasks to members, and making decisions with economic consequences.

Autonomous teams stimulate participation and involvement, and an effect of this increases emotional attachment to the organization, resulting in greater commitment, motivation to perform and desire for responsibility. As a result, employees care more about their work, which may lead to greater creativity and helping behaviour, higher productivity and service quality [12]. Self-management can also directly influence team effectiveness since it brings decision-making authority to the level of operational problems and uncertainties and, thus, increase the speed and accuracy of problem solving [22].

However, there is substantial variance in research findings regarding the consequences of such teams on such measures as productivity, turnover, and attitudes

[14, 22]. Tata and Prasada [22] found that employees really need to affect managerial decisions for achieving the benefits of a self-managed team. It is important that the team do not experience symbolic self-management.

Since the prioritizing of the backlog is a decision-rich problem solving activity, it is valuable to look at the classical theories of decision-making to better understand backlog decisions which will explain in the following section.

2.3. Decision-making

Our experience show that the introduction of agile development in a plan-driven organization leads to changes in decision making process which may lead to organizational changes. Since every project decision affects both the product and sprint backlog, we will focus on the processes related decisions in these backlogs when studying the decision-making processes in agile software development.

Although no theoretical perspective can fully explain all aspects of decision making process in teams, we have chosen to build on the work of Anthony [3] to investigate the behavioural aspects of decision makers in Scrum teams. Anthony [3] argues there are three levels of decision-making activities in organizations based on the purpose of the management activity: *strategic planning, management control and operational control*. As such, we view the decision making process in Scrum teams at the strategic, tactical and operational level.

We believe that in Scrum "strategic decisions" are primarily an organizational issue, "management control decisions" involve a project management view, whereas "operational control" decisions are about the implementation of features.

Several researchers discuss the alignment of managerial decisions in organizations and point out how important to ensure that software applications are well aligned with business strategies of organizations [4, 18]. Here we assert that for a team to be able to self-organize, decisions on all levels in managerial activities must be aligned. Hence, the first aim of this paper is to examine the alignment of decisions between the three levels i.e. strategic, tactical and operational level.

The introduction of self-organizing teams requires an agile team participating more in "management control" decisions. As explained in Section 2.2 Scrum-master works with the product-owner which leads a constant communication between the management control and operational control decision makers. Hence, we argue that even though the product-owner is responsible in prioritizing the backlog, the team must be involved in this process and their opinions must influence the final prioritization if they are to be able to self-organize. The second aim of this paper is to study the negotiation process between management control and operational control which requires intense communication between the product owner and Scrum teams.

The introduction of self-organizing teams also requires an agile team be responsible for all the "operational control" decisions. Hence, the third aim of this paper is to examine the decision making process at operational level among the Scrum team members.

3. Research Design and Methodology

The goal of this research is to provide insight into the decision making processes used in agile software teams; hence, it is important to study software development teams in practice. We will examine:

- Aligning the decision-making process on the three management levels i.e. strategic, tactical and operational level activities. We will also study the negotiation process, in particularly between the product owner and the Scrum-master.
- Decision-making on operational level activities.

We report on a multiple-case holistic study [24], in which we studied one phenomenon in two projects.

3.1. Study Context

This study was done in the context of a larger action research program, where several companies have introduced elements from agile development in response to identified problems. For the two companies where the projects in this study are taken from, Scrum was introduced because they wanted to improve their ability to deliver iteratively and on time, increase the quality, improve the team-feeling and team communication. One of the companies is situated in the south of the Norway and the other is situated in the northern part, therefore we name the project “South” and “North”. The first author of this paper participated in the introduction and training of Scrum, and observed the companies that were using Scrum.

3.1.1 Project South. This project was carried out within a medium-sized software company with approximately 150 employees in four departments. The goal of the project was to develop a system for integrity management of pipelines both off-shore and on-shore. Today several customers are interested in buying the product, and so far three contracts has been signed. One of the biggest challenges in this project is to align requirements from potential customers from all over the world. Scrum was introduced one year after the project had started.

The project consists of six developers in the core team working full time (one is a Scrum master), two GUI designers, one product-owner, and one project-manager devoting half his time on this project. The Project-manager is in charge of traditional project management tasks and visiting customers together with the Product-Owner. The sprints usually lasted three weeks, ending on a Friday with a retrospective- and review-meeting. The next sprint was planned the following Monday. The team organized a 15 minute stand-up every morning to discuss project related issues. The Product-owner and project-manager usually joined every Scrum-meeting.

Prior to Scrum, there was a problem in prioritizing what to develop first. The project involved developing a new system while concurrently trying to get input from the market department and potential customers, resulting in the team focusing on too many modules at the same time and not being able to deliver running software. The team had difficulties with prioritising their tasks as a result. At the same time of introducing Scrum there was a board-meeting deciding on what to develop first.

Before introducing Scrum, one person was responsible for the architecture. He was seen as one of the experts. However, he mostly communicated with the Product-owner and less with the team. The team became frustrated as they felt they were missing the big picture. The architect left the company during the first Scrum-phase.

3.1.2 North. In this company, the whole development department with 16 developers were introduced to Scrum through a two day workshop. This was the first Scrum project in the company. The goal of this project was to develop a plan and coordination system. The project produced a combination of textual user-interfaces and map-functionality. The company will also be responsible for maintenance and support after final installation. Six developers, 4000 hours, one Scrum-master, and a Product-owner were allocated to the project. The Product-owner was situated in another city. He acted as a representative for the customer who was the local government of a Norwegian city. The Scrum-master was a part of both the quality and marketing department, and has worked in the company for more than 10 years. The developers were highly specialised with corresponding division of work.

Scrum was introduced at the beginning of the project. The length of each sprint was one month – except for a 2 month summer-sprint. The sprint started with a planning meeting and ended with a review meeting, where the developed functionality was presented. Shortly after the review meeting, the next planning meetings were conducted. Several retrospective meetings were also organized, but since the product owner was situated in another city, he did not attend all of these. The team organized a 15 minute stand-up every morning. The Product owner participated in some of these by telephone.

3.2. Data Sources and Analysis

In order to understand how Scrum was applied and how the introduction of Scrum affected and changed the developing process, we conducted semi-structured interviews which lasted from 20 to 50 minutes. Everyone in the project was interviewed. We focused on understanding the coordination of work, communication in the team, feedback-sessions, planning and estimation, and how decisions were made. We also observed the teams, using ethnographic observations (Participant observation [15]), during the introduction of Scrum, and during several of the stand-up meetings. In this article we rely primarily on data from the interviews.

All data from this study was imported into a tool for analyzing qualitative data called Nvivo (www.qsrinternational.com). The first author coded the material, looking for material related to how decisions were made in the project.

4. Decision-making in Agile Projects

We now report our findings from the "South" and North" project, according to the previously identified aspects of decision-making.

4.1. Aligning Strategic, Management and Operational decisions

4.1.1 The South project. Prior to introducing Scrum there was no clear priority of what was most important and which feature had to be covered in the next release. The Scrum-master said: *after introducing Scrum and delivering every three weeks, it's easier to have a clearer focus on what to do. Everything can not be equally important. The product-backlog makes it*

easier for us to tell the Product-owner to prioritize what is most important. I think the problem is that he is talking to so many customers, and then his perspective constantly changes. The team perceived the prioritized backlog as a way of aligning all input to the project. However, combining the long-term and short-term perspective was still seen as challenging. One developer said: *Now we are using the product-backlog, and that is enough for coordinating work and aligning the input from every stakeholder. It is still a problem with too much interruption from the market. They ask us to think in a long term, but they often interrupt us with new input from the market. But the situation has improved. The sprint and sprint-backlog also makes it easier to say "no".*

Informing the team about the market-situation was still important, and this was done in the daily stand-up. The Project-manager said: *It is now easier to communicate what is happening in the market. After a meeting with a potential customer I always use 1-2 minutes informing the team about what happened in the next daily stand-up.* The team considered this feedback important.

However, exposing the team to new ideas had its drawbacks. For instance, a sudden change in scope by introducing new features during the sprint undermined the decisions made previously on the operational level as it made it difficult to deliver according to the sprint-plan. In order to prevent introducing new ideas to the team in the middle of a sprint, the Scrum-master organized separate meetings with the Product-owner and Project-manager preparing the backlog for the next sprint.

The Product-owner felt the process of discussing and making decisions together with the team was useful. While talking about the planning meetings he said: *We have good and deep discussions in these meetings when we are estimating the tasks and deciding on what to do. It's the person who knows most, that decides what to do.* The frequent dialogue also resulted in early identification of the problems at the operational and managerial level activities. The Product-owner said: *If the team get stuck, and do not know what to do because of missing or conflicting decisions, this is discovered early. Then we together decide what to do. Before Scrum, developers could use months before we discovered that there were problems we needed to discuss.*

Both the Product-owner and Project-manager had strong opinions of how work should be distributed in the team. The Product-owner argued that people should be specialised with corresponding division of work. There should be different people responsible for the DB, a GUI expert etc. He saw this as the most effective way of developing software. The Project-manager commented that: *They often make other plans than I would expect them to do. That is fair enough. When I present my idea of who should do what, they often say "now, we can't do it this way". Then they agree on how to do it. As long as they deliver what is expected I'm satisfied.* The Scrum-master said: *Often it is obvious who should do what. However, we need to think of the consequences by letting the database person only work on the database. We then get to vulnerable, and it will reduce his job-satisfaction. We need to let people work on the things that motivate them. This is something we have changed after introducing Scrum.* There was clear diversity in the perspective of how work should be coordinated, between the Scrum master and the project manager; however it was clear that the teams view was respected.

In contrast, the traditional form of work in combination with highly specialised developers resulted in a lack of understanding of the product by a significant portion of the team members. This made it difficult for aligning strategic management and operational decisions. One developer said: *Earlier the chief-architect took most of the decision himself. This was frustrating. Now we all participate. Also because of everything being in the backlog, everyone knows the long-term and short-term goal of the project. Earlier it was hidden in the architecture.* In the literature this is referred as the hold-up problem [13]. The introduction of agile approach allowed the team members to have the long term perspective about the product development.

4.1.2 The North project. The North project experienced different challenges. Although there was considerable communication by e-mail and telephone between the developers and the Product-Owner, the developers felt it was not sufficient, and that more was required to help them better understand how the system was going to be used. The geographical distance made the alignment process more difficult. In the planning meetings there were a lot of discussions about the intention of each feature and how the system was supposed to be used. It was obvious that this was important for being able to decide on what to do; however these discussions were time-consuming and seemed somehow to exclude the developers. The planning meeting usually ended in a discussion between the Scrum-master and Product-Owner. The team barely participated in the decision-making process at this level. It is thought that a lack of thorough discussions could be one of the reasons for important tasks being identified during and in the end of each sprint instead of at the beginning of each sprint. This again reduced the validity of the common

backlog, making the developers focus more on their own plan. Since the planning had weaknesses and none of the developers felt they had the total overview, this probably was one of the reasons for design-problems discovered later.

The company had not placed a customer-support process for software maintenance which caused a dilemma among the developers. If a developer was responsible for a module or system being developed, this developer would later be the one supporting the same solution. The developer would never know when the customer would make a call, and interrupt his/her work. The developers described this as the "quagmire". One of the developers said: *One of the problems with working here. If you ever get involved in developing a system you get stuck in the quagmire. And for every new project you are on, it gets worse.*

This approach was regarded as a competitive advantage by the company as they were fast handling the customer inquiries. However, these frequent interruptions annoyed the developers since they needed to find out themselves how to prioritize their resources. They were also allocated on several projects, so when there were problems in other projects this resulted in problems delivering according to the sprint plan. As one of the developers said: "*I do not like to be the one who decides which project not meeting the deadline"*. After introducing Scrum, the team-members felt that they were more protected against external noise and the interruptions were not as bad as it used to be but it was still a major problem.

4.2. Decision-making on an Operational Level

4.2.1 The South project. The developers usually coordinated themselves by the daily *stand*-up. This was the primary meeting for operational level decisions. The Scrum-master said: *When anyone gets trouble, and do not know what to do. We immediately discuss this in the team, and then together figure out what to do. If we can't decide in the team, I discuss with the Product-owner and the Project-manager. Last time we discovered that something was wrong in one of the modules, and that it would take 3 days to correct. The Product-owner then had to decide if we were going to solve the problem, or do it in the next sprint.*

In one sprint, one of the developers decided on to implement a feature without discussing with the other team members. The Scrum-master said: *We divide tasks among us, and then people are responsible for implementing them. This usually works fine. However, in the last sprint one of the developers spoke with someone in the market about an issue. The developer used a couple of days changing a feature that was already finished, because she thought this was very important. While testing I found out what had happened and we had to redo what the he developer had done.*

Although Scrum facilitated decision making among developers, this was not the case for some team members. One experienced developer newly hired on the project said: *They usually tell me what to do. I think I possess more knowledge than they realize, and that I get too simple tasks. I'm also missing a good overview of what we are going to deliver. Many of the discussion regarding the design were done without me, and I think there are several discussions I should have participated in. It's probably too expensive if everyone is to participate in every meeting, but for me it is difficult to see dependences between the modules. The module is documented, but it is not easy to understand.*

4.2.2 The North project. Each person worked on the module where he or she was seen as a specialist (e.g. user interface, map-interfaces or databases). There was a common understanding among managers and developers that this was the most efficient way of working. The scrum-master said: "*Let the person that knows most about the task solve it! It will take too many resources if several persons are working on the same module, and there is no time for overlapping work in this project. The tasks are delegated and solved the best possible way today. Maybe we can do more overlapping in the next project"*.

Because of the work being divided into modules, a developer typically created his own "plan" for this module, often without discussing it with the team. One of the developers decided to implement features for future projects without the knowledge of any other team member. As a result, this developer lost trust from his team and the Scrum-master started to give him direct instructions regarding work.

In many cases, module specific problems were seen as personal problems, and therefore they were not reported or discussed in the group. It was also difficult to discuss problems because developers did not know what the other team members were doing. This again resulted in some of the developer's not paying attention to what others were saying during the daily Scrum-meeting. One developer said: *When it comes to the daily scrum, I do not pay attention when Ann is talking. For me it is a bit far off what she talks about, and I do not manage to pay attention. She talks about the things she is working on. I guess this situation is not good for the project.*

The Scrum-master eventually became aware of problems not being reported. His strategy was to make the team quickly solve each problem, as soon as it was discovered, by paying more attention to the nature of

the problem and by discussing this within the team. However the developers felt the scrum-master was overreacting and creating a lot of noise as so began reporting even fewer problems. This again affected the decision process among the team members.

5. Discussion on Decision-making and Scrum

In this section we present our key observations in light of our research question:

How does decision-making theory explain the challenges involved in introducing Agile development?

In the South project we found that Scrum might help in aligning the decisions on every level, since Scrum requires a prioritize product-backlog. The team claimed it was useful to get constantly feedback from the market during the daily-sprint, but it was also identified as a problem protecting the Sprint-backlog against new prioritizing during the sprint. Here the team deviated from the Scrum theory, which argues that the customer can participate in the sprint meetings, but is not allowed to influence the team in between the meetings [20]. In the North project, the company had no appreciation of the problem as the strategic and management activities were misaligned and the developers needed to decide which project to work on. The developers were aware that the commitments they made in the planning-meting would change during the sprint. Hence Scrum did not work as intended.

On the operational level the South project experienced an improved situation, with everyone participating in the decision making since the architectural work was now divided among the team. They had eliminated their hold-up problem earlier on [13]. Problems were also discovered earlier and they were able to work on a prioritized backlog. However, even though the situation did improve an issue occurred where one of the developers felt they were not fully integrated in the team. In another similar case there were developers making decisions without informing the rest of the team – something referred to as decision hijacking [5]. In the North project there were problems with the decision-making process on the operational level. Team members felt that undertaking more responsibility in the decision-making process would result in maintenance work in the future which was not perceived as ideal. It was also difficult for the team to commit to the decisions made in the planning meeting because of the problems earlier described. In addition, the decision-making on the operational level was influenced by the highly specialised skills and the corresponding division of work since this resulted in the developers not knowing what the others were doing, and missing the total picture of the project The consequence of this was important task not being identified before late in the project and developers creating their own plan without discussing with others, and performing decision hijacking.

The problems in the North project with an ad-hock decision-making process on the operational level also affected the alignment process, since the developers perceived their own plan as more important than the total plan. This is also known as a lack of team orientation [19]. For every team, particularly for a self-organizing team, the team's goals should be more important then the individual goals.

6. Conclusion and Future Work

This paper presented data from a multiple case-study. We found that:

- Not appreciating the problem with miss-alignment of strategic and management activities is a barrier when introducing Scrum.
- Even though specialisation is important; this can result in a division of work, be a barrier for the decision-making process on the operational level, and a barrier to aligning decisions on the three management levels.
- Daily meetings are important for preventing decision-hijacking.
- Removing the hold-up problem makes it easier for developers to participate in the decision-making process.

Changing the way of working is difficult, and when it involves a transition from specialized skills to redundancy of functions, it requires a reorientation not only by the developers but also by management. This change takes time and resources, and it must be solved to be able to succeed with Scrum.

This study highlights several barriers with establishing a decision-making process, which is essential for succeeding when introducing agile development. Accordingly, further work should focus on investigating other barriers with introducing agile development. In addition, there is a need for more research on the decision-making process in mature agile teams.

7. Acknowledgment

We appreciate the input received from the investigated companies, and from Torgeir Dingsøyr who participated in the data collection. This research is supported by the Research Council of Norway under Grant 174390/I40.

8. References

[1] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, *Agile software development methods - Review and analysis*, VTT Publications, 2002.

[2] P. Abrahamsson, J. Warsta, M.T. Siponen, and J. Ronkainen, "New directions on agile methods: a comparative analysis," 2003, pp. 244-254.

[3] R.N. Anthony, *Planning and Control Systems: A Framework for Analysis*, Harvard University, Boston, USA, 1965.

[4] A. Aurum and C. Wohlin, "A Value-Based Approach in Requirements Engineering: Explaining Some of the Fundamental Concepts," in *Requirements Engineering: Foundation for Software Quality*, 2007, pp. 109-115.

[5] A. Aurum, C. Wohlin, and A. Porter, "Aliging Software Project Decisions: A Case Study," *International Journal of Software Engineering and Knowledge Engineering*, Publisher, city, 2006, pp. 795-818.

[6] K. Beck and C. Andres, *Extreme Programming Explained: Embrace Chage (2nd ed)*, Addison-Wesley, 2004.

[7] B.W. Boehm and R. Turner, *Balancing Agility and Discipline: A Guide for the Perplexed*, Addison-Wesley 2003.

[8] A. Cockburn and J. Highsmith, "Agile software development: The people factor," *Computer*, Publisher, city, 2001, pp. 131-133.

[9] D. Cohen, M. Lindvall, and P. Costa, "An Introduction to Agile Methods," in *Advances in Computers, Advances in Software Engineering*, vol. 62, Zelkowitz, M.V., Ed. Amsterdam: Elsevier, 2004.

[10] M. Cohn and D. Ford, "Introducing an agile process to an organization [software development]," *Computer*, Publisher, city, 2003, pp. 74-78.

[11] T. Dyba and T. Dingsoyr, "Empirical Studies of Agile Software Development: A Systematic Review," *Information and Software Technology*, Publisher, city, 2008.

[12] M. Fenton-O'Creevy, "Employee involvement and the middle manager: evidence from a survey of organizations," *Journal of Organizational Behavior*, Publisher, city, 1998, pp. 67-84.

[13] R.F. Freeland, "Creating holdup through vertical integration: Fisher Body revisited," *Journal of Law & Economics*, Publisher, city, 2000, pp. 33-66.

[14] R.A. Guzzo and M.W. Dickson, "Teams in organizations: Recent research on performance and effectiveness," *Annual Review of Psychology*, Publisher, city, 1996, pp. 307-338.

[15] D.L. Jorgensen, *Participant Observation: A Methodology for Human Studies*, Sage publications, Thousands Oak, California, 1989.

[16] S. Nerur and V. Balijepally, "Theoretical reflections on agile development methodologies - The traditional goal of optimization and control is making way for learning and innovation," *Communications of the ACM*, Publisher, city, 2007, pp. 79-83.

[17] S. Nerur, R. Mahapatra, and G. Mangalaraj, "Challenges of migrating to agile methodologies," *Communications of the ACM*, Publisher, city, 2005, pp. 72.

[18] L. Neumann-Alkier, "Think Globally, Act Locally – Does it Follow the Rule in Multinational Corporations? ," *5th European Conf. on Information Systems*, Cork, 1997, pp. 541-552.

[19] E. Salas, D.E. Sims, and C.S. Burke, "Is there a "big five" in teamwork?," *Small Group Research*, Publisher, city, 2005, pp. 555-599.

[20] K. Schwaber and Beedle, *Agile Software Development with Scrum*, Upper Saddle River: Prentice Hall, 2001.

[21] H. Takeuchi and I. Nonaka, "The New New Product Development Game," *Harvard Business Review 64: 137-146* Publisher, city, 1986.

[22] J. Tata and S. Prasad, "Team Self-management, Organizational Structure, and Judgments of Team Effectiveness," *Journal of Managerial Issues*, Publisher, city, 2004, pp. p248 - 265.

[23] L. Williams and A. Cockburn, "Agile software development: it's about feedback and change," *Computer*, Publisher, city, 2003, pp. 39-43.

[24] R.K. Yin, *Case Study Research: design and methods*, Sage Publications, 2003.