December 2006

A Case Study on the Applicability and Effectiveness of Scrum Software Development in Mission-Critical and Large-Scale Projects

Juyun Cho
Utah State University

YongSeog Kim
Utah State University

David Olsen
Utah State University

A Case Study on the Applicability and Effectiveness of Scrum Software Development in Mission-Critical and Large-Scale Projects

Juyun Cho
Utah State University
jcho@cc.usu.edu

YongSeog Kim
Utah State University
yong.kim@usu.edu

David Olsen
Utah State University
David.Olsen@business.usu.edu

ABSTRACT

Agile software development methods, including Scrum, Extreme Programming (XP), and Dynamic System Development Method (DSDM) have been mainly applied to small-scale and dynamically evolving software development projects (e.g., Internet and Web software development) with some success, due to their lighter and faster development life cycle. However, there has been very limited evidence of the effectiveness of such methods (in particular Scrum) in mission-critical and large-scale software development projects. This paper will present the findings of a case study on the applicability and effectiveness of Scrum in such projects. We first briefly summarized the findings on what aspects of Scrum were both successfully and poorly adopted (applicability perspective) and what aspects of Scrum were key factors to improve or deteriorate the quality of the products in terms of bug rates, development time and costs (effectiveness perspective). Based on literature reviews of small group research, we also analyzed the effectiveness of Scrum from the perspectives of social facilitation, social loafing, and group motivational gains. In addition, we identified five core characteristics of Scrum and analyzed the relationships among them. Finally, this paper provides some suggestions to help organizations overcome obstacles expected in adopting the Scrum method as a future development tool.

Keywords

Agile development, Scrum, autonomous team

INTRODUCTION

Recently, agile software development methods (such as Scrum, XP, and DSDM) have garnered a lot of attention from practitioners in the software engineering industry. This is mainly because of their potential to improve customer satisfaction, decrease defect rates, and shorten development time along with their capability to accommodate rapidly changing requirements (Boehm and Turner, 2004). However, most of these methods have been primarily applied to small-scale and relatively simple projects. It is not clear either whether the agile methods can provide the end-users with the desired quality in a timely manner on large-scale projects (Marrington, Hogan, and Thomas, 2005), although some researchers have reported that large-scale and complex projects have also benefited from suitably tailored agile development methods (Bowers, May, Melander, Baarman, and Ayoob, 2002; Cao, Mohan, Xu and Ramesh, 2004). Further, the majority of studies on large-scale projects have been conducted using the XP method which was initially designed for small-scale projects, with less than 10 developers and a product that would not be excessively complex (Beck, 2000). Although the Scrum method is claimed to be suitable for all project sizes (Schwaber and Beedle, 2002), there have been very limited studies on the applicability and effectiveness of Scrum in large-scale projects. Therefore, it is worthwhile to conduct research to find out whether the Scrum method is applicable to and effective for large-scale projects.

From the perspective of applicability, we first reviewed several key elements of Scrum and explained how each element was adopted to minimize the implementation risks. We also measured the effectiveness of Scrum in terms of bug rates, development time and costs. In addition, the concepts of social facilitation (Bond, 1982; Cottrell 1972), social loafing

(Harkins and Jackson, 1985), and group motivational gains (Balijepally, 2005) were used to explain the effectiveness of Scrum. We also noted the relationships among five core characteristics (autonomous team, efficient product management, enhanced communication, iterative and continuous monitoring, and human resource management). Finally, we addressed the net effects of these core characteristics on the quality of the products.

The remainder of this paper is organized as follows: Agile software development with Scrum is first introduced along with the most common elements of Scrum, and then the research methodology is explained. Next, we summarize the findings on what aspects of Scrum were successfully and poorly adopted (applicability perspective) and what aspects of Scrum were key factors to improve or deteriorate the quality of the products in terms of bug rates, development time and costs (effectiveness perspective). After that, we present managerial implications and the relationships among five key characteristics. We conclude the paper with some suggestions for possible future research.

AGILE SOFTWARE DEVELOPMENT WITH SCRUM

In this section, we briefly introduce the most common elements of Scrum based on the explanation from Schwaber (Schwaber et al, 2002), who formulated the initial version of Scrum development process. In general, the Scrum method is a team-based lightweight process to manage software development using iterative and incremental practices. The origin of the term, *Scrum*, can be found in a popular sport, rugby, in which two teams consisting of eight individuals compete against each other. While the term Scrum refers to the strategy used for getting an out-of-play ball back into play in rugby, it was used to describe the process of developing products in 1986 (Takeuchi and Nonaka, 1986). As in a rugby match, development teams in Scrum are organized to have holistic movement, have continuous interaction among team members, and have intacting core team members. Among the team members, *Scrum Master* (SM) is the person who does administrative work, such as arranging the daily Scrum meeting and location, and removing any production impediments. The SM also serves as a liaison between the team and other departments. The SM also keeps track of what is going on in daily Scrum meeting to gauge the velocity of the team.

We also noticed that there are several important meetings in Scrum including the *daily Scrum meeting, the daily Scrum of Scrums meeting, the sprint planning meeting,* and *the sprint review meeting*. The daily Scrum meeting is a short (usually 5-15 minutes) standup meeting in which developers talk about what has been done since last meeting, what will be done before the next meeting, and what the impediments are to progression. Note that each development team may have different schedules for the daily Scrum meeting. The daily Scrum of Scrums is a daily meeting for SMs from multiple Scrum teams. It is important to remember that both daily meetings are supposed to be a short standup meeting. In addition to these informal daily meetings, two monthly, more formal meetings are used. The sprint planning meeting is a monthly meeting where team members divide one of the items in the product backlog into a set of small and manageable tasks, which will be entered into the sprint backlog. The product backlog contains a prioritized list of all product requirements determined by the product owner. Based on the estimated completion time for each task, the set of all tasks in the sprint backlog is determined in such a way that they can be completed within a month. In this way, as small tasks are completed, developers can see the progress easily and can have a sense of accomplishment at every Scrum meeting. The main objective of another monthly meeting, the sprint review meeting, is to check what has been done, what things need to be improved, and what things the team has done well. Typically, it takes a day to conduct the sprint review meeting and the sprint planning meeting. An open working environment is recommended in Scrum, because it allows people to communicate more easily, makes it easier to get together, and facilitates self-organization.

We subjectively identified four core characteristics of Scrum as follows: autonomous team, efficient product management, enhanced communication, and iterative and continuous monitoring. First of all, the success of software development with Scrum is heavily dependent on how each team is constructed. In particular, the Scrum method allows developers to run the team and have more ownership on the projects that they are working on. Developers like the idea that the team decides how to do things based on consensus, and that the team has more control over how the development is managed and completed. Overall, the autonomous team in Scrum enables the developers to facilitate better team work and better communication that result in products of higher quality (Hanakawa and Okura, 2004). Note, however, that a team consisting of team members unwilling to help each other will significantly decrease product quality and require more time and costs for development. From the perspective of efficient product management, a product backlog and a sprint backlog are used to help developers prioritize tasks, and a burn down chart is used to keep track of task assignment. In Scrum, the priority of projects and task assignment to developers was dynamically changed based on the team members' progress on the task and business requirements. Another characteristic of Scrum is its iterative and continuous monitoring on the progress of the projects. The daily Scrum meeting and the sprint review meetings help developers monitor the progress of the projects iteratively and continuously. In particular, when there are high priority tasks, the daily Scrum meetings help developers stay on task and remind them what needs to be done on a daily basis.

RESEARCH METHODOLOGY

A case study was conducted for eight months at a company founded in 1978. This company has been developing and maintaining software for more than 500 clients across 36 states in USA. In particular, this company specializes in mission-critical and sophisticated public safety software, such as jail management, fire/emergency medical service (EMS) management, computer-aided E911 dispatch, mobile communications, and records management systems (RMS). These systems must meet the highest standard of robustness and reliability because of their great impact on public safety. For example, due to a computer glitch in the jail management system, seven inmates were released early last year in Michigan (Rosencrance, 2005).

The company in the current case study has a total of 30 software developers and QA personnel, which are divided into five development teams in the software division. To test the success of Scrum, about 25 software developers and QA personnel, including one of authors, were invited to two 8-hour intensive training sessions, which were conducted by one of the leading training experts in this field. After the training, the company decided to adopt Scrum and reorganized their development teams to fit into the Scrum method. Each team consisted of four to seven software developers, a Quality Assurance (QA) person, a SM, and a Product Line Manager (PLM).

Email surveys and interviews were conducted among software developers and QA personnel. In addition, one of authors observed how the Scrum method was adopted and implemented in the company while he was participating in a jail management project as a software developer. Ten developers responded to the email survey while five technical leads in each team, a QA manager, and a PLM were interviewed. The interviews and surveys contained three main questions: 1) What things did and did not work for you on a project? 2) What are the most interesting and unique aspects of Scrum? and 3) What did you learn, what would you do next time and what advices do you have for others? These questions served us well in giving information on the applicability and effectiveness of the Scrum method. The results of the interviews and the email survey are summarized and categorized in the following subsections.

APPLICABILITY AND EFFECTIVENESS OF SCRUM

The Applicability of Scrum

When Scrum was first introduced to the company, most developers were reluctant to adopt a new method because of their experiences in a traditional development method (e.g., waterfall method). However, they soon began to enjoy several unique features of Scrum as they exercised various elements of Scrum. In general, most elements of Scrum were successfully adopted and implemented with minor modification with regard to a meeting schedule and an open working environment.

The company first tried to keep the size of the Scrum team as small as possible because small teams can be more flexible and adaptable in defining and applying an appropriate variant of Scrum (Rising and Janoff, 2000). In terms of the daily Scrum meeting, a couple of teams had a standup meeting and the rest of teams had a seated meeting. However, the principal of short daily Scrum meeting time was well observed (less than 15 minutes) by all teams. Several developers claimed that it was not necessary to have Scrum meeting everyday when there were no specific agenda to discuss, but the majority of developers and QA personnel enjoyed this daily-based meeting. The company could not invite clients to this meeting because there are so many clients scattered in the United States. Instead, the PLMs visited the clients to show the progress of the products and to obtain feedback from the clients. For those clients with whom the PLMs could not visit, a web conference which enables up to 20 clients to watch online demonstration was used. The PLMs spent about a half of their work time talking with the clients using phone, email, or web demos as well as onsite visits. In addition, the company hosted users' conference once a year to get feedback from the clients. In this conference, clients voted for or against a new direction of application development. The feedback from the clients was discussed and reflected on the product backlog.

The company also appointed a SM for each one of five development teams. Due to the company's belief that the main role of SMs is to provide the administrative services for the team members, the company appointed all SMs from non-technical persons. A group of SMs held daily Scrum of Scrums meeting everyday for 5 to 15 minutes with company's vice president of software development division. The company also held a sprint review meeting called a "products fair". The products fair was unique in the sense that developers in all teams, QA personnel, and people in other departments (sales, marketing, customer support) were strongly encouraged to attend. The company expected that, given an opportunity to show their accomplishments during the sprint, developers would diligently work to make sure that they are currently on the pre-determined development schedule with appropriate levels of intermediate deliverables of the products. Note that sprint planning and review meetings are usually led by the SM or the PLM, while the PLM created the product backlog and communicated with clients to reflect their opinions on the product backlog. In order to improve the communication and interactions among team members, the company relocated all the scattered developers in the same team into the same place where two developers shared cubicles.

From the perspective of applicability, a small and manageable team, an efficient daily and monthly meeting, a competent SM, customer feedback, and a productive working environment are the key factors that should be adopted appropriately for the success of Scrum.

The Effectiveness of Scrum

In the first stage (first three months) of the projects, the Scrum method was not very effective in terms of bug rates. For example, during the first three months, the QA department has found almost twice as many bugs since the company switched to the Scrum method. They attributed this finding to the complexity of the project. As the size and complexity of the application grew, the dependencies and interconnections among tasks in the application increased. However, the developers were not able to fully consider all the dependencies and interconnections among modules because of their myopic planning and design in each sprint planning meeting (Elssamadisy and Schalliol, 2002). Through personal interviews with developers, we also found that when developers had to complete a set of tasks determined in each sprint planning meeting, they had a tendency to complete tasks in a quick and dirty way rather than to think of how each task will be flexible enough for future needs. Further, the developers often found that their monthly work schedules to produce intermediate deliverables were optimistically estimated. As a result, they failed to finish all the tasks in the sprint backlog. After the company supported and encouraged the developers to spend more time considering the dependencies and interconnections among modules, and the flexibility of the code, bug rates gradually decreased to the normal rates. However, we did not see evidence that the Scrum method lowered normal bug rates.

In terms of development times and costs, Scrum was not very effective during the first two months after the company switched to the Scrum method. We attributed this finding to the composition of development teams. The teams were reorganized without considering the knowledge and skills of the developers. Therefore, some team members had to learn business logic (how the application works in a specific field) that they were not familiar with, and others needed to learn new development tools and programming languages. As a result, it took a longer time and more costs. However, once they became familiar with business logic and mastered new development tools and programming languages, development time and costs were reduced.

In order to further improve the effectiveness of the Scrum method, the company used two different management tools: Microsoft Excel and VersionOne. Microsoft Excel was first used, but was replaced by VersionOne because of its limited capability. The VersionOne is a web-based commercial management tool that provids various functionalities such as simplifying project planning and management, enhancing business and project adaptability, improving project visibility, and increasing project predictability and confidence.

Social Facilitation, Social Loafing, and Group Motivational Gain

Social facilitation indicates that in the presence of other people, performance is facilitated on a simple task, whereas performance on a difficult task is hampered (Aiello and Douthitt, 2001). Social loafing is the tendency to take advantage of others' efforts when working in groups, while group motivational gain can be obtained when people increase their effort to help co-workers whose performance is poor.

Social facilitation effect and group motivational gains were observed as factors that reduced development times and lowered bug rates. In particular, evaluation apprehension (Bond, 1982; Cottrell, 1972) was observed when the company invited all of developers, QA personnel, and people in other departments (sales, marketing, customer support) to the sprint review meeting. Because in the sprint review meeting, all of developers had a chance to show what they had done in an interdepartmental environment, they were concerned about how they were evaluated by these people. Group motivational gains were also obtained as a social compensation effect (Williams, Harkins and Karau, 1991) when a team member helped other team members who were not familiar with new development tools or programming languages. At the same time, the company did not have ways to evaluate individual performance, leaving room for individuals seeking to free-ride. This social loafing should be eliminated by making individuals' contributions verifiable (Balijepally, 2005) to offset negative impacts on development time and costs.

THE MANAGERIAL IMPLICATIONS

In this section, we would like to address relationships among key characteristics of the Scrum method and managerial insights to maximize the benefits of software development based on Scrum. In addition to four core characteristics (autonomous team, efficient product management, enhanced communication, and iterative and continuous monitoring) that we identified in a previous section, we include human resource management to reflect the importance of team composition on the success of software development. In terms of human resource management, the company should consider what would be an optimal allocation of limited number of developers. It was obvious that it took more development time and cost when new team members did not interact well with other members of the team (Williams and Kessler, 2000) or if team members needed more training to complete their tasks. The company should also carefully examine the role of a QA person in each Scrum team. Currently, each Scrum team has a QA person to test the codes, relieving the burden of all developers from testing their own codes. However, this causes a new problem: a QA person could test only parts of modules that the developers finished by the end of the sprint and, as a result, another QA test must be scheduled in the following sprint. Note that QA personnel are always behind in the current sprint because they find bugs in the code that developers threw together in the previous sprint. Assigning an appropriate SM is also important for the success of the Scrum method. Sometimes team impediments were not taken care of, and the impediments slowed or even stopped the progress of the sprint. If developers need to do other projects asked by other departments they cannot focus on the sprint backlog items and Scrum will not work best. In conclusion, human resource management can directly affect the performance of Scrum teams.

The autonomous nature of the Scrum team allows the developers to have more control over how and when development is completed, depending on the consensus of the team, and have more ownership of the projects that they are working on. Therefore, it is tempting to state that an autonomous Scrum team is ideal for efficient product management. Note, however, that developers are not often able to fully consider all the dependencies and interconnections among modules, resulting in inconsistent product outputs across team members within the same Scrum team. Due to the limited communication among different Scrum teams, it is difficult to keep the output of product consistent across teams. For example, several teams may simultaneously work on "look and feel" design that should be implemented by a designated team (e.g., an architecture team). This implies that there should be a person (possibly a SM in each team) who is responsible for checking consistency of products across Scrum teams (Marrington et al, 2005). Therefore, we found that the inconsistency and duplicates across different teams could negatively affect the efficient product management.

It is clear that the autonomous nature of the Scrum team substantially increases the communication among developers within a team through daily and monthly Scrum meetings. In particular, the daily Scrum meeting is a good time for team members to understand what other members are working on and identify obstacles to overcome. For example, if a developer would say, "I am stuck with a certain problem", other developers would typically respond, "I can help with that. Let's get together after the meeting", or, "I don't know the solution for that but I can look at your problem after the meeting". Through good communication, team members are able to refine the goal for each sprint and improve the quality of products (Hanakawa et al, 2004). However, it is noted that while sharing cubicles with a coworker can improve the opportunity to communicate among team members, it would be possible for developers to be constantly distracted by cubicle partners who are often having conversations with other coworkers. An interesting field study shows that teams are more productive when their members work independently (Parrish, Smith, Hale, and Hale, 2004). Further, it may not be a good idea to have developers run the team without having a team leader (Marrington at el, 2005) to make a right decision in a timely manner when team members do not know which way to take among several alternatives.

The autonomous nature of the Scrum team also creates an incentive to monitor the progress of the projects iteratively and continuously because it is the developers who create and update the priority of tasks through interactions and discussions in the daily Scrum meeting and the sprint review meeting. Further, the various features of the management tool for these meetings are also beneficial in the sense that they help developers visually remember what needs to be done on a daily basis. Note also that the iterative and continuous monitoring of the projects is tightly coupled with efficient product management. For example, daily scrum meetings can be considered a good way to both continuously monitor the projects and efficiently manage the quality of the product.

We finally summarized our findings in Figure 1 to show the interactions among the five factors and their directions using arrow heads. The plus and minus signs indicate positive and negative influences respectively. We found that these five factors should interact positively when Scrum was applied to large-scale and mission-critical projects.

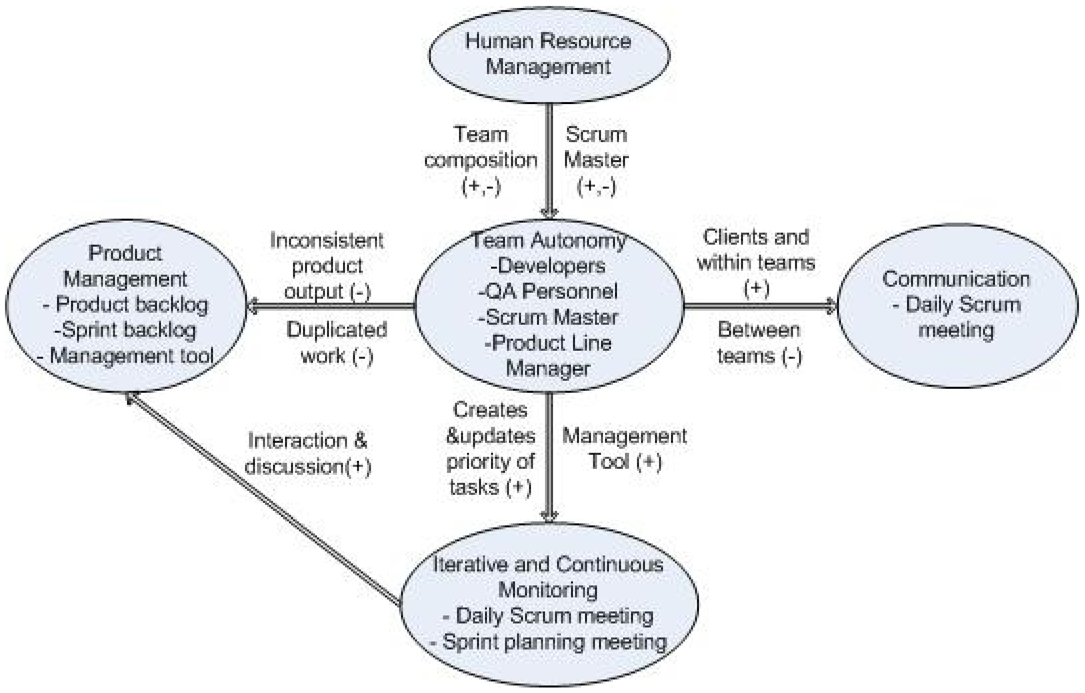
Human Resource Management
Team composition (+,-)
Scrum Master (+,-)
Product Management - Product backlog -Sprint backlog - Management tool
Inconsistent product output (-)
Duplicated work (-)
Team Autonomy -Developers -QA Personnel -Scrum Master -Product Line Manager
Clients and within teams (+)
Between teams (-)
Communication - Daily Scrum meeting
Interaction & discussion(+)
Creates &updates priority of tasks (+)
Management Tool (+)
Iterative and Continuous Monitoring - Daily Scrum meeting - Sprint planning meeting

Figure 1. Interactions among the five factors.

CONCLUSION

The need to develop a robust and reliable high quality software product within a short period of time has forced many companies to adopt agile software development methods; most companies applied such methods mainly to small-scale and relatively simple projects with some success. However, it is questionable whether such methods could bring the same success when they are applied to mission-critical and large-scale projects. This paper presents findings from a case study on the applicability and effectiveness of Scrum on mission-critical and large-scale projects. We summarized the findings as follows:

- Compose teams after considering each team members' knowledge and skills on the projects that they will work on. Without the right mix of team members, Scrum process becomes much slower and more expensive.
- Designate a SM who can not only remove administrative and technical impediments of the team but can also manage the overall product consistency across teams.
- Allocate enough time for planning and design to efficiently manage large and complex projects.
- Tailor the Scrum method depending on the nature of applications and the development environment because there is no single software development process that is the best approach for every project (Little, Greene, Phillips, Pilger, and Poldervaart, 2004; Huo, Verner, Zhu, and Baber, 2004).

The lessons learned in this case study could be valuable to other organizations that are already using Scrum or planning to implement it in the future. This case study was conducted on on-going projects so it would be worthwhile to examine how much development time and costs are needed to fix the bugs reported by the clients after the final products are delivered. Due to numerous clients scattered across the nation, communication with customer was done through PLMs and this was quite limited. Better models should be studied for this situation to get feedback from clients more efficiently. We also briefly examined social facilitation, social loafing, and group motivational gain. We think further studies on the effectiveness of Scrum are needed with this viewpoint. Finally, we hope further case studies will be published on the applicability and effectiveness of Scrum on mission-critical and large-scale projects to find the common success factors in adopting the Scrum method.

REFERENCES

1. Aiello, J. R., and Douthitt, E. A. (2001) Social facilitation from Triplett to electronic performance monitoring, *Group Dynamics*, 5, 3, 163-180.
2. Beck, K. (2000) Extreme programming explained: Embrace change, Addison-Wesley, Reading, MA.
3. Balijepally, V. (2005) Collaborative software development in agile methodologies – Perspectives from small group research, *Proceedings of the Eleventh Americas Conference on Information* Systems, August 11-14, Omaha, NE, USA.
4. Boehm, B. and Turner, R. (2004) Balancing agility and discipline: Evaluating and integrating agile and plan-driven methods, *Proceedings of the 26th International Conference on Software Engineering,* 718-719.
5. Bond, C.F. (1982) Social facilitation: A self-presentational view, *Journal of Personality and Social Psychology,* 42, 1042-1050.
6. Bowers, J., May J., Melander, E., Baarman, M. and Ayoob A. (2002) Tailoring XP for large systems mission critical software development, *Proceedings of the Second XP Universe and First Agile Universe Conference on Extreme Programming and Agile Methods – XP/Agile Universe 2002,* August, 100-111.
7. Cao, L., Mohan, K., Xu, P. and Ramesh B. (2004) How extreme does extreme programming have to be? Adapting XP practices to large-scale projects, *Proceedings of the 37th Hawaii International Conference on System Sciences (HICSS'04),* May 15-21, Hawaii.
8. Cottrell, N.B. (1972) Social facilitation, in C.G. McClintock (Ed.) *Experimental Social Psychology,* Holt, New York, 185-236.
9. Elssamadisy, A. and Schalliol, G. (2002) Recognizing and responding to "bad smells" in extreme programming, *Proceedings of the 24th International Conference on Software Engineering*, May 19-25, Orlando, FL, USA, ACM Press, 617-622.
10. Hanakawa, N. and Okura K. (2004) A project management support tool using communication for agile software development, *Proceedings of the 11th Asia-Pacific Software Engineering Conference (APSEC'04), 316-323*
11. Harkins, S.G. and Jackson, J.M. (1985) The role of evaluation in elimination social loafing. *Personality and Social Psychology*, 11, 575-584.
12. Huo, M., Verner, J., Zhu, L. and Babar, M. (2004) Software quality and agile methods, *Proceedings of the 28th Annual International Computer Software and Applications Conference (COMPSAC'04),* v01, IEEE Computer Society, 520-525.
13. Little, T., Greene, F., Phillips, T., Pilger, R. and Poldervaart, R. (2004) Adaptive agility, *Proceedings of the Agile Development Conference (ADC'04),* June 22-26, 63-70.
14. Marrington, A., Hogan, J. M. and Thomas, R (2005) Quality assurance in a student-based agile software engineering process, *Proceedings of the 2005 Australian Software Engineering Conference (ASWEC'05),* March 29-April 1, Australia, 324-331.
15. Parrish, A., Smith, R., Hale, D. and Hale, J. (2004) A field study of developer pairs: Productivity impacts and Implications, *IEEE Software,* 21, 5, 76-79
16. Rising, L. and Janoff, N. (2000) The scrum software development process for small teams, *IEEE Software*, 17, 4, 26-32.
17. Rosencrance, L. (2005) Computer glitch gives seven Mich. Prisoners early, http://www.computerworld.com/governmenttopics/government/story/0,10801,105707,00.html.
18. Schwaber, K. and Beedle, M. (2002) Agile software development with Scrum, Prentice Hall, Upper Saddle River, NJ.
19. Takeuchi, H. and Nonaka, I. (1986) The new new product development game, *Harvard Business Review*, 64, 1, 137-146.
20. Williams, K.D., Harkins, S.G., and Karau, S.J. (1991) Social loafing and social compensation: The effects of expectations of co-worker performance, *Journal of Personality and Social Psychology,* 61, 4, 570-581.
21. Williams, L and Kessler R. (2000) All I really need to know about pair programming I learned in kindergarten, *Communication of the ACM,* 43, 5, 108-114.