Quantifying the Effect of Using Kanban vs. Scrum: A Case Study[1]

Dag I.K. Sjøberg, *University of Oslo*

Anders Johnsen and Jørgen Solberg, *Software Innovation*

Abstract

Many claims about the usefulness of various processes or methods, such as Scrum and Kanban, have been stated in agile and lean software communities. However, these claims are rarely supported by objective data or empirical investigations. In contrast, this article aims to demonstrate that the effect of processes or methods (here: Scrum versus Kanban) can be evaluated and compared on the basis of objective data. We analyzed data of more than 12,000 work items collected over the years 2009-2011 in a medium-sized software company. The company used Scrum from 2007 to autumn 2010, at which point they changed to Kanban. By using Kanban instead of Scrum, the company almost halved the lead time, reduced the number of weighted bugs by 10%, improved productivity by 21% for PBIs, and reduced productivity by 11% for bugs. Consequently, Kanban seems to outperform Scrum in this company. However, the results should be interpreted with caution because the use of Kanban succeeded the use of Scrum. To acquire more knowledge about the performance of different agile or lean methods, scholars should conduct similar studies in different organizations in different application domains and with people of different cultures and competences.

Introduction

Suppose that you are a manager in a software development company that has used Scrum for some period of time. You feel that Scrum, primarily due to its timeboxing, is too rigid for your company. You wonder whether it would be better to use Kanban than Scrum. However, first you would like to see some evidence that by introducing Kanban, your company will show improved performance with respect to the critical criteria, such as lead time, quality, and productivity. At present, you would not be able to find such evidence reported in the literature.

Most people would agree that empirical evidence collected in a systematic manner should be part of the basis for important decisions. Data that can back up claims is considered necessary in most scientific or engineering disciplines. However, in the IT industry, there is generally little solid evidence for the utility of a method in a given setting. Collecting relevant data is perceived to be too difficult and resource-demanding for most software organizations, and academia seems not to have prioritized this area.

Empirical studies have been conducted on certain agile practices [1], in particular, pair programming [2], but we have not found any scientific study with industry data that compares the effects of using various agile or lean methods on the ultimately interesting variables, such as lead time, quality, and productivity. In response, we report such a study here.

[1] This is a preliminary version of the article that will appear in a special issue on Lean Software Engineering, *IEEE Software* Sep./Oct. 2012.

Setting

Software Innovation (SI) is a Scandinavian company that has developed and sold document management products for 28 years. These products are built on the Microsoft SharePoint platform and are tightly integrated into the Microsoft Office environment. At present, approximately 100 developers and specialists work in 10 teams on the products. In total, SI has 330 employees distributed over five countries. The developers and testers are mainly located in Oslo, Norway and Bangalore, India. SI has partners in 12 countries and 400 customers.

From 2001 to 2006, SI followed a waterfall process with an annual cycle of design, implementation, testing, and deployment for each new release. In the beginning of 2007, the company carefully examined its development process, which resulted in a decision to introduce Scrum. Scrum was implemented with standard elements, such as cross-functional teams, sprint planning meetings that included estimation of work items using planning poker, daily standup meetings, sprints of three weeks with shippable increments of code (fully tested) at the end of each sprint, and demos in the review meetings. The status of the work was made visible through automated reports and task boards for all of the teams.

After a couple of years, the second author (R&D Operations Manager in SI) and third author (CTO) felt that Scrum was too rigid, did not scale, and was unsuitable for maintenance tasks. They also feared that the combination of inaccurate estimates and timeboxes gave longer lead times and that both timeboxes and what they perceived as "waste", such as Scrum planning meetings, reduced productivity and quality. Therefore, in 2010, the company switched from Scrum to Kanban.

Kanban is implemented at SI in the following manner. When work has started on an item, the company attempts to let the item flow through all of the stages until it is ready for release at a satisfactory quality as soon as possible (fast delivery), that is, without using timeboxes. Furthermore, only a limited number of work items are in progress at the same time (WIP limit). If the WIP limit has been reached, work will not start on a new item before another one is finished (just-in-time). Another change from the Scrum period is that SI no longer finds that they need cross-functional teams. Start-up meetings with estimation of work items have been abandoned. When running Kanban, SI still organizes daily standup meetings, but instead of demo meetings at the end of each sprint, status meetings with demos are held once or twice a week, regardless of the progress of the work items being discussed. There is no difference in the quality gates between Scrum and Kanban; all code is equally shippable.

TABLE 1. MEASURING PROCESS QUALITY

	Name	Values
Independent variables	Process	Scrum or Kanban
	Type of work item	Bug or Project Backlog Item (new features, adaptive maintenance tasks, and support tasks, i.e., all tasks that are not bug fixing)
Control variable	Year.Quarter	Each quarter from 2009.1 to 2011.4
	Churn	Number of lines added, deleted, or modified
Dependent variables	Lead time	Number of days from "Next" state to "Ready for release" state on the board
	Production	Number of work items developed per quarter (often called "throughput" [3])
	Productivity	Production per developer
	Productivity 2	Total churn per developer per quarter
	Quality	Number of weighted bugs in the severity levels: Blocking (weight 8), Critical (4), Moderate (2), and Minimal (1)

In 2011, the second and third authors realized that "being rigorous with agility just because it is written in theoretical books showing toy examples is of no business value" [4, Ch. 8]. Therefore, they contacted the University of Oslo to help evaluate their hypothesis that the company had benefitted from switching from Scrum to Kanban with respect to lead time, quality, and productivity. To investigate this hypothesis, we (primarily the first author) analyzed data on more than 12,000 work items that was collected over three years (2009-2011) by using Microsoft's Team Foundation Server (TFS). Although the company started using Scrum in 2007, we analyzed data only from 2009 onwards because SI had overcome the start-up difficulties with agile development by this point. Table 1 shows the independent, control, and dependent variables used in this study.

Lead Time

The Merriam-Webster dictionary defines *lead time* as (1) "the time between the beginning of a process or project and the appearance of its results". The Collins dictionary provides two definitions: (2) "*manufacturing*: the time between the design of a product and its production" and (3) "*business*: the time from the placing of an order to the delivery of the goods." For a consultancy company contracted with a customer who requests tailored software solutions, definition (3) is a useful starting point; that is, lead time may be defined as the amount of time between the proposal of a new feature or another request and its deployment in the customer's environment.

However, for an in-house development company such as SI that provides two-three releases of its products a year to 400 customers, definition (3) is unsuitable for two reasons. First, the amount of time a work item remains in the backlog queue before it is put on the board is a function of priority, not whether the company uses Scrum, Kanban, or other development methods. Furthermore, companies that develop and sell products to many customers may propose new features themselves and put them on the backlog *before* any customer requests them. Second, given a policy of two-three

releases a year, the result of a work item is not delivered to the customer immediately after it has been finished.

Consequently, definition (1) above is more appropriate; that is, we define lead time as the amount of time that passes from the moment that the development team receives a request to the moment that the team completes the work item. This definition is consistent with the one given in [5]: "the time for an item to move all the way across the board."

Figure 1 shows the average lead time for bugs and PBIs for each quarter within the periods in which SI used Scrum or Kanban. In the 3rd quarter of 2010, both Scrum and Kanban appear in the data because during that period, some teams in SI still used Scrum, while other teams had switched to Kanban. (To prevent outliers from having a large effect, we removed the work items with the 10% longest lead times in each quarter for each type of work item. As a result, the analysis set was composed of 10804 work items.)

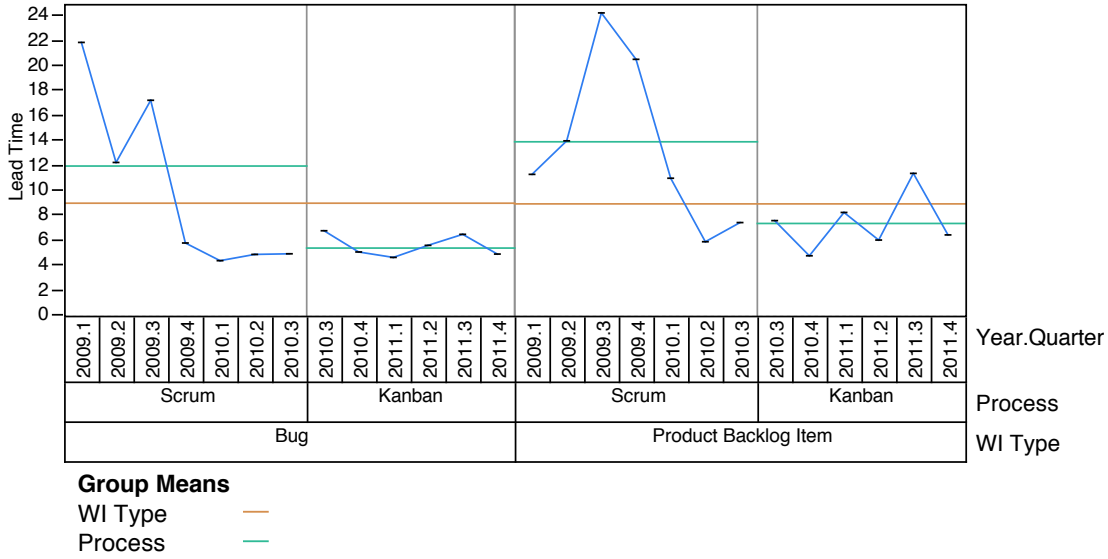

Figure 1. Average lead time measured in days by work item type, process, and quarter

The figure shows that the average lead time declined by approximately 50% from the Scrum period to the Kanban period. For the bugs, the average lead time fell from 12 days for Scrum to 5 for Kanban. For the Project Backlog Items (PBIs), the lead time declined from 14 to 7 days. The orange line in Figure 1 indicates that the bugs and PBIs had the same average lead time (9 days) over the whole period. The local top on each 3rd quarter is due to less activity in the summer holiday. The figure does not show the large variation in lead times. The standard deviation in the Scrum period (17 days for bugs and 20 for PBIs) was much greater than the standard deviation in the Kanban period (5 days for bugs and 9 for PBIs). As the figure shows, the long lead times for Scrum occurred in 2009. In 2010, the lead time of the Scrum period was at the same level as the lead time in the Kanban period (from 2010: 4.7 days for Scrum bugs vs. 5.4 for Kanban bugs, 8.2 days for Scrum PBIs vs. 7.4 for Kanban PBIs).

Churn

A change in development performance may be due to aspects other than a change in the formal process. For example, this change could be due to changes in the products and the technological environment. However, in this case, both products and environment have been very stable. Furthermore, this work assumes that the average amount of work per work item is stable over time. We do not have timesheets that show how many hours each developer or tester has spent on each work item. Instead, we use churn as a surrogate measure of effort. Churn is defined as the sum of the number of lines added, deleted, and modified in the source code. A study with exact measures of effort found a correlation of 0.6 between effort and churn for the modification of existing files and a correlation of 0.7 for the development of new files [6].

Figures 2a and 2b show the average churn for bugs and PBIs, respectively. The average churn for bugs is 6% *higher* in the Kanban period than in the Scrum period, while for PBIs, the average churn is 12% *lower* in the Kanban period than in the Scrum period. We removed the work items with the 10% largest churns within each quarter for each work item type before conducting the analysis to reduce the effects of outliers. This analysis pertains to the work items that involved changing code, which comprised approximately half of all work items (i.e., those with churn > 0). This finding indicates that the size of the work needed to finish a work item may change over time, although the changes are not dramatic. Only a small correlation exists between churn and lead time at the individual file level (for bugs, Spearman's ρ = 0.13, $p < 0.01$; for PBIs, Spearman's ρ = 0.17, $p < 0.01$). However, at the quarterly level, a medium, non-significant correlation exists between average churn and average lead time for bugs (ρ = 0.45, p = 0.15), while a large, significant correlation exists for PBIs (ρ = 0.71, p = 0.01).

Consequently, even if we account for the possible changes over time in the effort needed to finish a work item, as measured by change in churn, the average lead time still declines by approximately 50% from the Scrum to the Kanban periods (58% for bugs and 40% for PBIs).

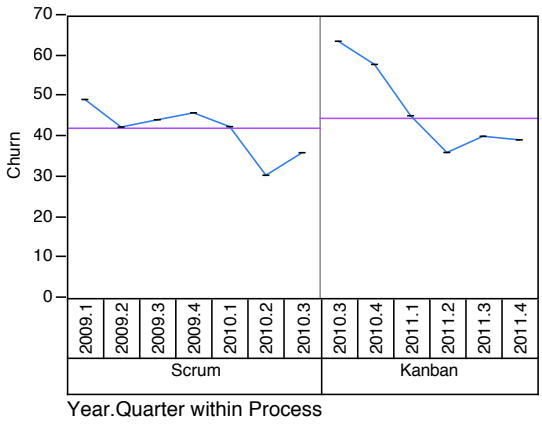

Figure 2a. Average churn of bugs

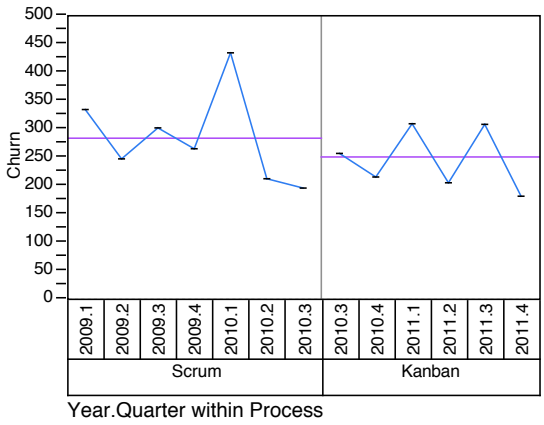

Figure 2b. Average churn of PBIs

Quality

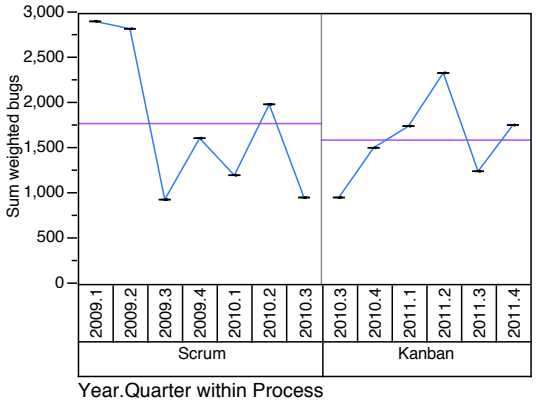

Figure 3a. Weighted bugs

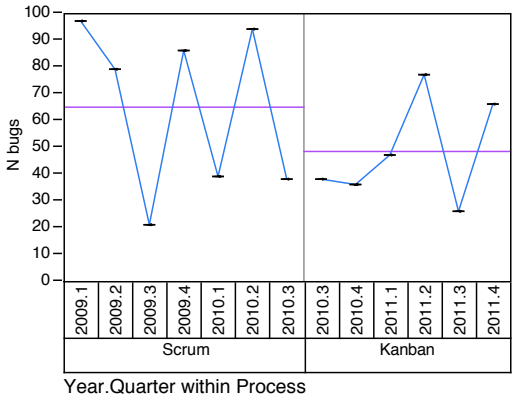

Figure 3b. Blocking bugs

According to the ISO/IEC standard 9126, a software system has six major dimensions that pertain to quality: functionality, reliability, usability, efficiency, maintainability, and portability. In this study, we focus on reliability, which is important because bugs in an operational system may lead to undesirable outcomes, such as system crashes or corruption of data. To measure reliability, we used the number of bugs, which were classified into four levels of severity, as indicated in the Orthogonal Defect Classification [7]. We gave each bug a weight corresponding to its level of severity; see the last row of Table 1. In SI, bugs are detected both internally (70%) and externally by SI's customers (30%). Most of the internal bugs are detected the last three weeks before a release because of intense manual and automatic testing in that period.

Figure 3a shows that the average number of weighted bugs per quarter fell from 1774 in the Scrum period to 1591 in the Kanban period (i.e., 10%). The variability declined much more; that is, the standard deviation was 832 for Scrum and 476 for Kanban. The most critical bugs, blocking bugs, declined in number even more between the two process periods (i.e., from 65 to 48 (26%); see Figure 3b). The standard deviation fell from 31 to 19. The dip in the third quarters is mainly due to less bug fixing during the summer holiday.

Figure 3a shows that more weighted bugs were found in the Scrum period during the two first quarters of 2009. Afterwards, Scrum was no worse than Kanban. Hence, the reduction in the number of bugs may be independent of whether the process was Scrum or Kanban. In any case, these numbers must be interpreted with caution. An increase in the number of bugs *may* be due to better bug detection or larger products. Since 2009, SI has employed more and presumably better testers, and the code base of their three products is being continually extended.

Production and Productivity

We measure production in terms of the numbers of bugs fixed and PBIs finished. The number of bugs fixed is almost the same over the Scrum (mean per quarter 595, stddev 271) and Kanban periods (mean 580, stddev 164), whereas the production of PBIs has more than tripled from the Scrum period (mean 190, stddev 50) to the Kanban period (mean 601, stddev 227).

However, one can usually increase production by employing more people. In the long run, productivity may be more important to a company than its total production. In SI, the number of developers and testers who fixed bugs increased from an average of 40 in the Scrum period to 48 in the Kanban period. The number of people who worked with PBIs increased from 34 to 59. The productivity (i.e., the number of work items per person) decreased from 15.3 to 12.1 (21%) for bugs (Figure 4a) but increased from 5.9 to 10.2 (73%) for PBIs (Figure 4b).

By using churn as an indicator of work item size (see Figures 2a and 2b), we propose an alternative measure of productivity to validate the above results. Specifically, we define Productivity 2 as the total churn divided by the number of developers in each quarter. Figure 5a shows that for bugs, the productivity decreased from an average of 0.46 KLOC (stddev 0.22) for Scrum to 0.41 KLOC (stddev 0.12) for Kanban (i.e., a reduction of 11%). Figure 5b shows that for PBIs, the productivity increased from an average of 1.28 KLOC (stddev 0.39) for Scrum to 1.55 KLOC (stddev 0.61) for Kanban (i.e., an increase of 21%). Consequently, if we adjust for work item size measured by churn, there is a reduction in productivity for bugs, but productivity still increases considerably overall from the Scrum period to the Kanban period.

The productivity gain in the Kanban period should also be viewed in light of the growth in the number of employees and the reduction in the number of project managers. In a period during which the number of employees increases, one would usually expect the productivity per employee to decline slightly because of organizational and communication overhead [8]. Furthermore, despite almost doubling the number of developers and testers, SI managed to reduce the number of (costly) project managers from four to three by transitioning from Scrum to Kanban.

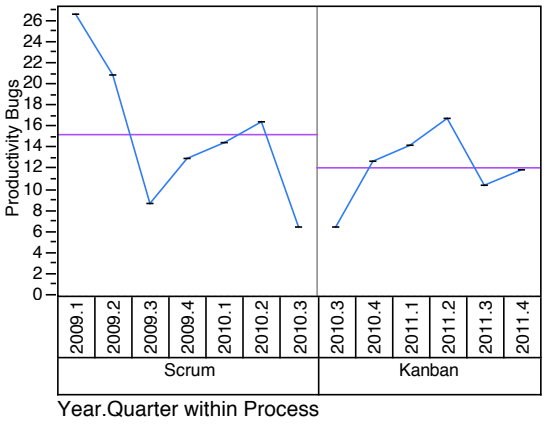

Figure 4a. Productivity: bugs per developer

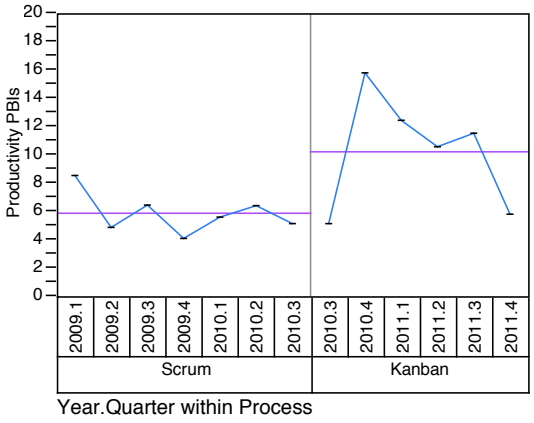

Figure 4b. Productivity: PBIs per developer

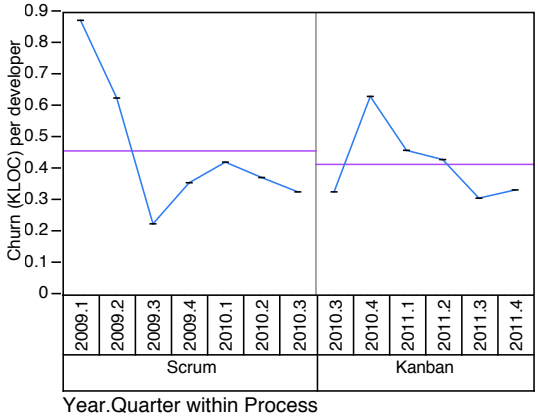

Figure 5a. Productivity 2: bugs per developer

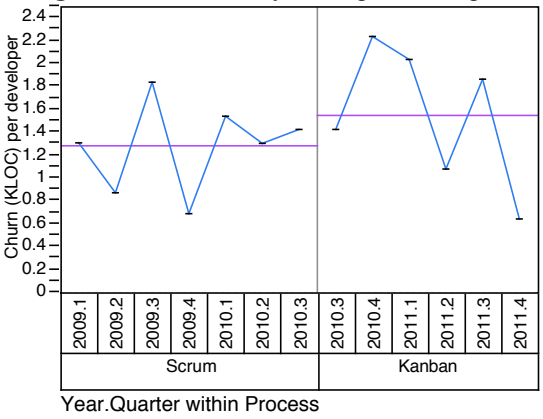

Figure 5b. Productivity 2: PBIs per developer

Qualitative Evaluation

To complement the quantitative data presented above, we report the opinions of the R&D Operations Manager (second author), CTO (third author), one team leader, and one developer, all of whom have been with SI for more than ten years. The team leader and developer were interviewed for one hour each by the first author. All these people clearly favored Kanban over Scrum.

The fixed timeboxes in Scrum were perceived to be artificial. The work items were frequently underestimated, and the developers also had to deal with ad hoc bug fixing, support, and maintenance tasks while working on the items. Still, one was supposed to finish the items within the given timebox. In practice, this timeline led to work items that were finished before the quality was satisfactory, that were deferred to the next iteration (which required new planning activities), or that were not finished at all. In the Kanban period, at least all of the items that had been started were finished because the developers focused on one item at a time until it was finished.

Furthermore, it was difficult to allocate the resources optimally within the sprints. For example, the testers tended to have little to do in the beginning of a sprint and too much at the end. Much of the sprint start-up meetings were perceived as "waste". In fact, SI had already reduced the sprint planning activities (and abandoned cross-functional teams) by the end of 2009. This relaxation of the Scrum rules was mentioned by two of the employees as an explanation for why the lead time was reduced from 2009 to 2010 (Figure 1).

Did the lack of timeboxes in Kanban lead to insufficient pressure to finish items? The consensus stated that the combination of daily stand-up meetings and weekly status meetings, the visibility of the items' status on the board, and the personal ambitions to complete the job constituted sufficient pressure.

Is Kanban better than Scrum?

As reported above, after replacing Scrum with Kanban, SI almost halved its lead time, reduced the number of weighted bugs by 10%, and improved productivity. Consequently, SI appears to benefit more from using Kanban than from using Scrum. Therefore, we recommend software companies that face difficulties with effort estimation and interruptions caused by ad hoc bug fixing, support, and maintenance tasks to consider using the lean practice of Kanban.

Nevertheless, as for any kind of study in software engineering, generalizing the results of case studies is challenging. Even though SI had been using Scrum for almost two years before the data analyzed in this study was collected, much of Kanban's indicated advantage may have simply been due to the fact that Kanban was used after Scrum. However, SI had become familiar with agile methods (Scrum) over more than three years before Kanban was introduced, and other aspects, such as SI's technological environment and products, were basically the same in both the Scrum and Kanban periods. The readers should judge for themselves whether they are in a situation that is similar enough to this company to apply the results of this case study into their environment.

To provide the agile and lean software community with more evidence on how various processes, particularly Scrum and Kanban, work for different organizations or teams in different contexts (e.g., application domains, cultures, and competences), we encourage other companies to collect and analyze data similar to our dataset. However, collecting high-quality data may be a challenge. Obtaining reliable information about the performance of a particular process or method requires reliable raw data. In a hectic environment, companies may find it difficult to motivate their developers and testers to record information continually about the states of the work items on which they are working. However, our experience suggests that people become motivated if they observe that the data that they record lead to useful feedback. In addition to feedback on the overall effects of various processes, SI also displays information about the number of bugs detected in the last week and month on monitors in common areas of the company's building. When visiting the company, partners and customers can then observe the number and trends of bugs in the various products.

Our study compared Scrum with Kanban. However, different implementations of these processes might have given different results, which is another reason why our study should be replicated in other environments. For example, a particular characteristic of Kanban is that the work in progress (WIP) should be limited, but Kanban does not specify what the WIP limit should be. To test the effects of various WIP limits, we plan to conduct a controlled experiment in which some teams will have lower WIP limits than other teams. We will then measure the team performance based on the same success criteria described in this article.

Acknowledgement

This work was partly funded by the Research Council of Norway through the project TeamIT, grant 193236/I40. We thank the interviewees at Software Innovation for participating in this study.

References

1. T. Dybå and T. Dingsøyr. Empirical studies of agile software development: A systematic review. *Information & Software Technology*, 50(9-10):833–859, 2008.
2. J.E. Hannay, T. Dybå, E. Arisholm, and D.I.K. Sjøberg. The Effectiveness of Pair-Programming: A Meta-Analysis, *Information and Software Technology* 55(7):1110-1122, 2009.
3. D. Anderson. Kanban: Successful Evolutionary Change for Your Technology Business. Blue Hole Press, April 2010.
4. C. Ebert and R. Dumke. *Software Measurement*. Springer, New York 2007.
5. H. Kniberg and M. Skarin. *Kanban and Scrum – making the most of both*. InfoQ, Lulu.com, 2009.
6. D.I.K. Sjøberg, A. Yamashita, B. Anda, A. Mockus, and T. Dybå. Quantifying the Effect of Code Smells on Maintenance Effort. Submitted for publication in *IEEE Trans. Software Eng*. 2012.
7. R. Chillarege, I. Bhandari, J. Chaar, M. Halliday, D. Moebus, B. Ray, and M.Y. Wong. Orthogonal Defect Classification—A Concept for In-Process Measurement, *IEEE Trans. Software Eng.*, 18(11):943-956, Nov. 1992.
8. T.K. Abdel-Hamid and S.E. Madnick. *Software Project Dynamics: An Integrated Approach*. Englewood Cliffs, New Jersey, Prentice Hall. 1991.