An Empirical Study of WIP in Kanban Teams

Dag I.K. Sjøberg
Department of Informatics,
University of Oslo, Norway
and
SINTEF Digital
Trondheim, Norway
dagsj@ifi.uio.no

ABSTRACT

Background: Limiting the amount of Work-In-Progress (WIP) is considered a fundamental principle in Kanban software development. However, no published studies from real cases exist that indicate what an optimal WIP limit should be. **Aims:** The primary aim is to study the effect of WIP on the performance of a Kanban team. The secondary aim is to illustrate methodological challenges when attempting to identify an optimal or appropriate WIP limit. **Method**: A quantitative case study was conducted in a software company that provided information about more than 8,000 work items developed over four years by five teams. Relationships between WIP, lead time and productivity were analyzed. **Results**: WIP correlates with lead time; that is, lower WIP indicates shorter lead times, which is *consistent* with claims in the literature. However, WIP also correlates with productivity, which is *inconsistent* with the claim in the literature that a low WIP (still above a certain threshold) will improve productivity. The collected data set did not include sufficient information to measure aspects of quality. There are several threats to the way productivity was measured. **Conclusions**: Indicating an optimal WIP limit is difficult in the studied company because a changing WIP gives contrasting results on different team performance variables. Because the effect of WIP has not been quantitatively examined before, this study clearly needs to be replicated in other contexts. In addition, studies that include other team performance variables, such as various aspects of quality, are requested. The methodological challenges illustrated in this paper need to be addressed.

ESEM '18, October 11–12, 2018, Oulu, Finland
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5823-1/18/10…$15.00
https://doi.org/10.1145/3239235.3239238

CCS CONCEPTS

• **Software and its engineering** → Software development process management

KEYWORDS

Agile and Lean development, quantitative study, lead time, productivity

1 INTRODUCTION

Kanban has become one of the most widely used methods in agile systems development. One of the salient principles of Kanban is the limit on Work-In-Progress (WIP), that is, the number of work items that are in progress concurrently. According to Kniberg and Skarin [1], a Kanban board is similar to a Scrum board, except that a Kanban board has a number that indicates the maximum number of work items that are allowed to be in progress at a workflow state at any time. Anderson et al. [2] define the Kanban software process as “a WIP limited pull system visualized by the Kanban board.”

1.1 Claimed Benefits of WIP Limit

Most proponents of Kanban (researchers, consultants, bloggers, practitioners, etc.) claim benefits of limiting WIP that include both efficiency and quality. Limiting WIP is supposed to make it easier to obtain a continuous flow of the whole process since the risk of bottlenecks and queues is reduced. Continuous flow is a principle of Lean that is central to Kanban. Prioritized work items should be worked on until they are finished, independent of the time it takes, which is in contrast to the principle of timeboxing in Scrum. By reducing the number of items that a team works on concurrently, more effort will be devoted to each item, which will then be delivered faster; that is, lead time is reduced. Little’s law [3] from queuing theory is often used to illustrate this relationship. It states that $L = \lambda W$, where L = average number of items in the queuing system, W = average waiting time for an item in the system, and λ = average number of items arriving per unit time. If we replace L with *WIP*, W with *Lead time*, and λ with *Productivity*, we get*: WIP = Productivity*Lead time;* or *Lead* *time = WIP / Productivity*. Accordingly, for a given productivity value, less WIP means shorter lead time.

More attention to fewer items may also increase quality by ensuring sufficient time for design, testing, etc. Putting fewer items on the board also encourages a more careful prioritization process and helps prevent overproduction.

Even though there are benefits to having a WIP limit that is not too high, there are also problems if it becomes too low. It may be impractical when many team members work on the same item, and productivity will be reduced if many members are idle because they have to wait to work on an item until others have finished their work on that item.

1.2 What Should the WIP Limit Be?

Kanban has been used in software development for about a decade [4], but to my knowledge, no quantitative, empirical studies have been reported on the real effect of different values of WIP. I have found only a couple of simulation studies [5], [6]. Several case studies on Kanban in general have been published (see, for example, the systematic review in [7]), but those studies only state that WIP limit was introduced as part of implementing Kanban, without any particular investigation on the effect of various WIP values. Another systematic review concludes that there is generally "a lack of reported scientific research addressing Kanban usage in software engineering" [8].

The absence of published studies may be the reason that, despite all the recommendations for reducing the WIP limit, no proposals exist in the literature for what the limit actually should be. According to Kanban proponents, you have to find out yourself by experimenting in your own environment. There is some guidance on where to start, though. For example, Kniberg and Skarin [1] suggest starting with *2n–1*, where *n* is the number of team members. (The "–1" is to prevent cases where everyone is working on two particular items all the time.) Then the WIP should be varied and the effect observed on "things like capacity, lead time, quality, and predictability" [1]. But to my knowledge, no published studies exist on the experience of teams that have conducted such experiments. The absence of relevant studies in the IT industry makes it difficult for managers and practitioners who adopt Kanban to determine the WIP limit.

1.3 Motivation for this Paper

Given the importance of WIP limits claimed in the Kanban literature, I wanted to investigate whether the WIP in the teams of a company that has been using Kanban for several years actually had an effect on team performance. If a variation in WIP results in a corresponding variation in performance, then the WIP limit should be defined close to the optimal WIP found empirically. The first research question is thus:

RQ1: Which WIP optimizes team performance?

In addition to the optimal WIP value that we may find in this study, another contribution of this paper is the discussions and illustrations of how to measure the effect of WIP. While other companies or researchers may collect the same kind of data, they may find other optimal WIP values and thus define other WIP limits because of context variables that may affect optimal WIP and thus WIP limits. Yet others may take our approach as a starting point but may find other data and analyses to be more appropriate in their contexts, which leads to the second research question:

RQ2: What are the challenges of quantifying the effect of WIP limit on the performance of Kanban teams?

The remainder of this paper is organized as follows. Section 2 describes the research method, which includes the studied company, data collection and a detailed discussion of the concepts and variables used in the study. Section 3 reports the results. Section 4 interprets the results and outlines their implications for practice and research with respect to the two research questions. Section 5 concludes.

2 RESEARCH METHOD

2.1 Background of the Case Study

The object of this study was the Scandinavian company Software Innovation (SI). The background for selecting this company was that the CTO and head of R&D attended a seminar organized by the Norwegian ICT interest group, in which I gave a presentation on earlier studies on software processes. After the presentation, they contacted me to initiate a collaboration where I should take the lead on analyzing the data they collected on their development processes.

SI has developed and sold document management products for more than 30 years and has more than 400 customers. These products are built on the Microsoft SharePoint platform and are tightly integrated into the Microsoft Office environment.

In 2007, SI replaced a waterfall process with Scrum, and in 2010, Scrum was replaced by Kanban, which is implemented as follows. After work has begun on an item, it flows 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. SI organizes daily standup meetings and holds demos once or twice a week. More information about SI and their development processes can be found in [9].

2.2 Data Collection

To investigate the effect of different WIP values, we collected data stored in the company's software management tool, Microsoft's Team Foundation Server (TFS). We analyzed 8,505 work items that were developed by the five teams over a period of 3.5 years (from the second half of 2010, when Kanban was introduced, until the end of 2013). The data collected about a work item includes what type it is (bug or feature); severity level if it is a bug (blocking, critical, moderate or minimal); to which product the work item belongs; which team is involved; who is working on that team; the date for each change of state (in progress or done for each the states analysis, development, test, and finally "ready for release"); links to related source files; and

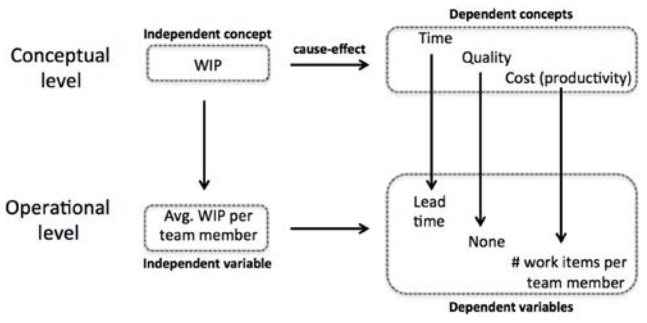

Figure 1: Conceptual model with "independent" and "dependent" constructs.

code churn (the number of lines added, modified or deleted in the source code).

2.3 Conceptual Model

Software development in an industry setting needs to consider time, quality and cost. It is therefore common to evaluate a process, method, technique, practice or tool with respect to its effect on these three dimensions, known as the "project triangle" (or "iron triangle" or "magic triangle"). To make empirical studies in software engineering as realistic as possible [10], such studies should attempt to investigate the effect of the studied phenomenon along these three dimensions. However, one of the most difficult challenges in empirical software engineering is to provide measurements that represent the concepts of interest. A concept that is well defined and operationalized is called a *construct*. *Construct validity* is the extent to which the definition is precise and the operationalizations span and represent the concept [11]. Fig. 1 illustrates the relationship between the concepts of this study and their operationalizations, which will be described below.

A particular challenge in this study is that the collected data does not include measurements of the size of a work item, which is the core unit of analysis. If a potentially large variation over time of the size of the work items is not controlled for, the variation would compromise some of the analysis reported below. To provide some control of the size, I used code churn.

2.3.1 Code Churn. In the period I studied, both the products and the environment of the company were very stable, but the size (in terms of effort) of the work items may have changed over time. No timesheets show how many hours each developer or tester spent on each work item. Instead, I use code churn as a surrogate measure of effort. In an earlier study, where we had exact measurements of effort, we 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 [12].

When using churn in the analyses, I removed work items that did not involve changing code (i.e., those with churn = 0), which comprised 31% of all work items. Before conducting the analysis, I also excluded the work items with the 10% largest churns within each quarter for each team, to reduce the effects of cases where whole modules had been deleted and cases where large chunks of code were added that included code copied from elsewhere. In such cases, churn does not adequately reflect the amount of work.

2.3.2 WIP. The overall motivation of this study is to help managers and practitioners in implementing the Kanban principle of limiting WIP. While I could have studied the effect of various WIP *limits* in different teams, it makes more sense to study the effect of varying WIP itself before one puts certain limits on it.

One may consider WIP at the full board level or at the level of each of its states, where different states can have different WIP limits. Even with access to information about more than 8,000 work items, I considered the number to be too low to defend analyzing WIP at the state level. Dividing the analysis into the nine commonly used states and a few other occasionally used states would have made the analysis more complicated, and the statistical power would have been reduced. Therefore, in this study, WIP is defined as the number of work items in progress, independent of whatever intermediate states on the board the items actually are in.

The raw data provided by TFS enables the calculation of WIP each day. By passing through all dates in the period of study (3.5 years), WIP is increased by one if a new work item is put into progress and reduced by one if an item is ready for release (see [13] for details). However, to study the effect of WIP on time and cost, for example, a time period of one day would be too short. In accordance with what we decided in another study that compared the use of Scrum with Kanban in the same company [9], a time period of one quarter seems appropriate. Hence, the WIP used in the analysis of this study is calculated as the daily WIP on average over a quarter. This study covers 14 quarters.

In Kanban, a WIP limit is specified at the team level, not at the individual level. The capacity of a team to manage different WIP sizes clearly depends on the size of the team (larger teams will generally manage higher values of WIP than smaller teams). To study the effects of different WIP values, it therefore makes sense to normalize the WIP with respect to the size of the teams. The independent variable, as shown in Fig. 1, is thus defined as the average WIP in a quarter divided by the average number of team members in that quarter.

Nevertheless, identifying team size was not trivial in this study, because several people worked in more than one team in the same quarter. In the absence of timesheets, I calculated a person's contribution to a team as the relative amount of churn of the tasks that were assigned to that person; that is, if the churn of the tasks was 400 in Team A and 100 in Team B, the person's contribution was 0.8 in Team A and 0.2 and in Team B. However, I could not use churn for the work items that did not involve a change of code. If at least one of the work items assigned to a person who worked in several teams in a given quarter had zero churn, I calculated a person's contribution relative to the number of items they worked on; for example, if a developer worked on 6 items in Team A and 2 items in Team B, the share would be 0.75 in Team A and 0.25 in Team B.

On average, 17.6% worked in two teams; only 0.2% worked in three teams, so an inaccurate measure of contribution to each team may not seriously influence the validity of this study. However, such an inaccuracy is still a threat to validity.

2.3.3 Time. The dependent variable *Lead time* adheres to the definition given in [1]: “the time for an item to move all the way across the board.” More precisely, lead time is defined as the number of days from the “Next” state to the “Ready for release” state on the board.

An alternate definition of lead time, the period of time between the proposal of a new feature (or another request) and its deployment in the customer’s environment, would not make sense for an in-house development company such as SI, which provides two to three releases of its products a year to hundreds of customers. A work item may be ready for release long before it is actually released, and a request for a feature may not be implemented until many customers or a particularly important customer requests it.

2.3.4 Quality. Job performance is typically a combination of time and quality [14]. A given amount of work may be conducted faster if one sacrifices quality, and vice versa. Ideally, this study should have used one or more indicators of quality to primarily study the effect of WIP on quality and to secondarily control for a potential increase or decrease of quality over time. For example, if the quality of the developed work items were reduced over time, one would expect shorter lead times and increased productivity.

However, no straightforward measure of quality was available. A manual inspection of the quality of each of the more than 8,000 items was not feasible. While I could have used the number of bugs at different severity levels, which are already stored in TFS, it is difficult to know to which period of time (quarter) the bugs belong. A bug may be produced in one quarter, detected in another, and fixed in a third; the company does not record systematically when a bug was made or detected.

As a consequence of the absence of a quality measure, quality was assumed to be constant and thus not to interfere with the studied effect of WIP on lead time and productivity. Although there are no indications that quality changed over time, this assumption is certainly a threat to the validity of the study.

2.3.5 Cost (Productivity). In all industrial software development, costs must be taken into account. A cost-benefit analysis of replacing one technology (process, method, technique, practice or tool) with another one must not only compare the running costs and benefits of the competing technologies but must also consider transitional costs, such as the training of personnel and even the recruitment of new people with other skills. Nevertheless, changing a WIP limit is straightforward and has scarcely any transition costs. So, in this case, a change in costs would be determined by a change in productivity.

Productivity is a function of how much is produced of what quality in what period of time and using how many resources. Quantity is in this study measured in terms of number of work items. However, as described in Section 2.3.1, the collected data does not describe the size of the work items, but we will use churn to adjust for a potential change in work item size over time within a team; for example, if X work items were produced in both Quarters 1 and 2 but the average churn was 10% higher in Quarter 2, the quantity of production increased by 10% in Quarter 2. However, the churn measure applies to only 62% of the work items. (As described in Section 2.3.1, 31% of the work items had churn = 0, and we removed 10% of the remaining items with longest churn, that is, 69% minus 10% (of 69) = 62%.) It would be unreasonable to assume that the change in size of the 38% of work items that are not included in the churn calculations is the same as for those that did have a churn measure. I therefore weighted the changed churn relative to the proportion of work items that are involved with churn measures; that is, a change in work item size was only calculated for 62% of the work items.

Generally, quality should be included in a measure of productivity. Unfortunately, as described in Section 2.3.4, there is no good measure of quality in this study, so it is regarded as constant.

The dominant cost in software development is salaries of personnel. The size of a team is thus an important factor when measuring software productivity. In summary, productivity is measured as the number of work items, adjusted by a churn factor, that a team makes ready for release within a quarter divided by the number of team members.

3 RESULTS

3.1 Lead Time

Fig. 2 shows the relationship between WIP per team member (from now on only called WIP), lead time, and productivity, calculated as the means over all quarters within a year and over all teams. The data shows that an increasing WIP (increasing from 2.8 in 2010 and 3.1 in 2011 to 4.1 in 2012 and 4.0 in 2013) corresponds to a longer lead time (increasing from 7.8 days in 2010 to 9.3 days in 2013). While there is a high correlation between WIP and lead time (Spearman's $\rho = 0.80$), the correlation should not be considered statistically significant due to the small sample size. A nonparametric test of correlation is used because the variables do not follow a normal distribution.

When increasing the granularity of time period from year to quarter, the relationship between WIP and lead time disappears; see Fig. 3 and Table 1 ($\rho = 0.12$, $p = 0.69$).

Figs. 4a to 4e in the appendix show graphs of the relationships for each team. Tables 2a to 2e show the corresponding correlations. Only Team A shows a significant correlation between WIP and lead time ($\rho = 0.78$, $p < 0.01$).

The relatively high number of correlations (18) in Tables 1 and 2 suggests that the level of significance of the individual tests should be adjusted to reduce the probability of a Type I error (rejection of a true null hypothesis). The Bonferroni adjustments or less conservative methods, such as the Holm-Bonferroni method, could be used [15]. However, given the exploratory nature of this study and the increased probability of a Type II error (failing to reject a false null hypothesis), along with other problems of using adjustment methods [16, 17], the significance level was not adjusted in this study.

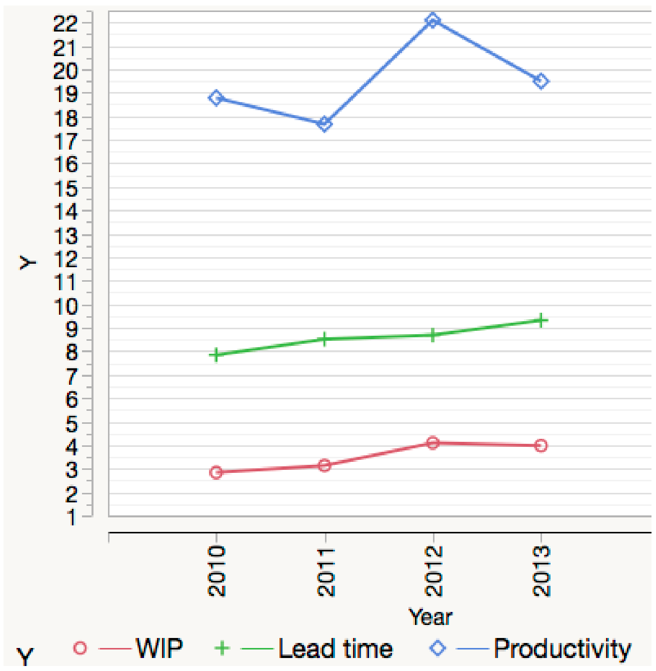

Figure 2: WIP, lead time and productivity by year.

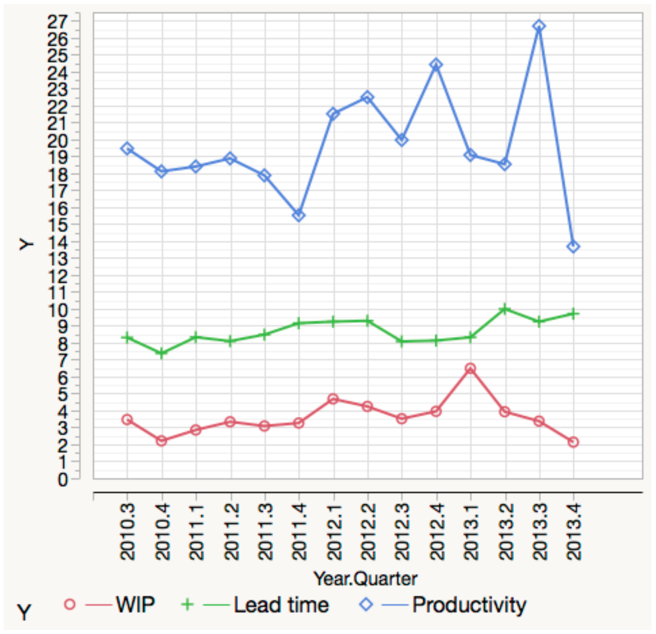

Figure 3: WIP, lead time and productivity by quarter.

Table 1: Overall correlations

Variable	by Variable	Spearman ρ	Prob>|ρ|	-,8-,6-,4-,2 0 ,2 ,4 ,6 ,8
Lead time	WIP	0,1165	0,6917	
Productivity	WIP	0,7143	0,0041*	
Productivity	Lead time	-0,0813	0,7823	

Table 2a: Correlations Team A

Variable	by Variable	Spearman ρ	Prob>|ρ|	-,8-,6-,4-,2 0 ,2 ,4 ,6 ,8
Productivity	WIP	0,3007	0,3423	
Lead time	WIP	0,7762	0,0030*	
Lead time	Productivity	-0,0140	0,9656	

Table 2b: Correlations Team B

Variable	by Variable	Spearman ρ	Prob>|ρ|	-,8-,6-,4-,2 0 ,2 ,4 ,6 ,8
Productivity	WIP	0,5077	0,0638	
Lead time	WIP	0,1297	0,6586	
Lead time	Productivity	-0,3582	0,2085	

Table 2c: Correlations Team C

Variable	by Variable	Spearman ρ	Prob>|ρ|	-,8-,6-,4-,2 0 ,2 ,4 ,6 ,8
Productivity	WIP	0,3934	0,1640	
Lead time	WIP	0,1429	0,6261	
Lead time	Productivity	-0,3846	0,1745	

Table 2d: Correlations Team D

Variable	by Variable	Spearman ρ	Prob>|ρ|	-,8-,6-,4-,2 0 ,2 ,4 ,6 ,8
Productivity	WIP	0,7413	0,0058*	
Lead time	WIP	-0,5245	0,0800	
Lead time	Productivity	-0,3846	0,2170	

Table 2e: Correlations Team E

Variable	by Variable	Spearman ρ	Prob>|ρ|	-,8-,6-,4-,2 0 ,2 ,4 ,6 ,8
Productivity	WIP	0,6868	0,0095*	
Lead time	WIP	0,2857	0,3440	
Lead time	Productivity	-0,0604	0,8445	

3.2 Productivity

Fig. 2 shows that the level of productivity has a minimum of 17.7 work items (adjusted for churn) developed per quarter per team member in 2011 and a maximum of 22.1 in 2012. There is a high correlation between productivity and WIP ($\rho = 0.80$), but note the small sample size also in this case.

When increasing the granularity of time period from year to quarter (Fig. 3), the high correlation between productivity and WIP remains and is clearly statistically significant ($\rho = 0.71$, $p < 0.01$). There is almost no correlation between productivity and lead time ($\rho = –0.08$, $p < 0.78$).

At the team level (Tables 2a–2e), there are significant correlations between productivity and WIP in Team D ($\rho = 0.74$, $p < 0.01$) and Team E ($\rho = 0.69$, $p < 0.01$).

4 DISCUSSION

This section discusses the results and methodological challenges according to the two research questions formulated in the introduction.

4.1 Which WIP Optimizes Team Performance?

The relationship between WIP and lead time found at the level of years and quarters (across all teams) accords with the claimed benefits of limiting the amount of WIP. Consequently, companies that are struggling with lead times should consider reducing WIP through a tougher prioritization of items that are brought from the backlog into progress.

The reason a clear relationship between WIP and lead time was present in only one of the five teams may be related to the *law of large numbers*; that is, when the sample size is large, the average of the results will be close to the expected value. The number of items is 2,126 per year on average and 608 per quarter. The average of 122 per quarter per team may be too small to show an effect. At the team level, presumably many other context variables affect lead time more than WIP, such as size and complexity of the items, teamwork quality [18] and individual skills [19].

The literature claims that development becomes more efficient with reduced WIP as long as it is above a certain threshold (e.g., 1 per person). (The average WIP level in this study (3.6) is clearly above such a threshold.) The finding in this study is in contrast to that claim. At the quarterly level (Fig. 3) and for two of the teams (Fig. 4), there are significant *positive* correlations between WIP and productivity. In other words, the teams seem capable of developing *more* items per unit of time (quarter) if WIP is high. It may be that having many items on the board provides the team with a certain pressure or sense of urgency that outweighs such effects of a high WIP as bottlenecks or frequent switching between tasks.

One may question why the relationship between WIP and productivity is not present at the overall year level (Fig. 2). Is the relationship not to be expected in the long run, and therefore the law of large numbers is not applicable; that is, are the positive correlations found constrained to particular cases, or are they spurious? Future studies are needed.

Advice to practitioners from this study would thus not be conclusive. If a short lead time is a priority, a low WIP limit seems sensible. If productivity is more important, a relatively high WIP limit should be defined.

4.2 What are the Challenges of Quantifying the Effect of WIP Limit on the Performance of Kanban Teams?

I have found only one study on WIP limits in software development. The researchers state: "Kanban focuses on limiting the amount of work in progress and visualizing work on the board. The team was satisfied with trying to keep the 'work in progress' at a low level because they felt that when they managed to do this it made them more effective" [20].

In contrast, this study attempts to obtain quantitative and objective data on how to implement the WIP limit requirement of Kanban. I do not know why no similar studies have been published. Perhaps is it too difficult; are the threats to the validity of the results one may obtain too many and too serious? I outlined several threats to construct validity in Section 2. For example, operationalizing the concept of team size was difficult. I used churn as a measure of contribution in each team when a person worked in several teams in the same quarter, but when there was no code change, I used the number of work items in the respective teams that were assigned to that person. However, a threat to that approach is that the size of the items most likely would be different only by chance, and it is likely that even the average item size varies across teams. I could have adjusted the expected size according to the variation in average churn size between the teams (the smallest average churn size of work items was 73 and the largest one was 131). But is it reasonable to suppose that the size of the churn in items that involve change of code reflects the size of items that do not involve change of code?

Instead of operationalizing the concept of amount of work into only one variable (number of work items) and thus facing the threat of mono-operation bias, churn could also have been an additional variable for measuring the overall amount of work. However, as stated above, not all work items involved code change, and while parts of some work items do include code change, those work items may also include substantial sections that do not contain changes to code.

The absence of a good measure of the size of a work item is also a threat to internal validity. The measurements of both lead time and productivity assume that variation in work item size is controlled for. I have adjusted according to churn when churn was nonzero and when it was not extremely high, but I have little evidence for how good churn is as a surrogate measure in this study. Timesheets might have been a better indicator of work item size, but that measure also has challenges. Timesheets may be useful for a given team over a short period of time, but in a longitudinal study like this one, which records data over 3.5 years, people may improve their work performance over time (i.e., produce more and/or better in the same time). Teams will also have turnover, and the large variation in individual work performance [19] will also affect the use of timesheets as a measure of item size. Nevertheless, timesheets were not used in the company I studied.

Furthermore, it is difficult to know how applicable the results from this particular study are to other companies and organizations. A large range of context variables will affect the external validity [21].

This study used data that was already being collected in the company before the study was designed. The shortcomings of the data described above can be mitigated if one carefully plans which data is needed and ensures that it will be collected. However, just asking a company to start recording timesheets for the purpose of a research question (even though it is relevant for the company) when they do not see the need for it otherwise is unlikely to succeed. One could pay the company for extra data collection, which we did in a multiple case study [22], but the duration of the software projects in that study was only a couple of months, and we had access to an extraordinary amount of available research money at the time. Another alternative is to identify companies that already record timesheets—for example, consultancy companies—and collaborate with them in research.

This study could also serve as a starting point for a well-designed, controlled experiment that explicitly tests the cause-effect relationships suggested in this correlation study. Shortcomings of this study could be addressed in such an experiment, but scale and realism would remain a challenge.

5 CONCLUSIONS

In this study based on a comprehensive data set from an agile company, a low WIP seemed to reduce lead time, which is positive and consistent with current claims in the literature and among leading practitioners. However, a low WIP also seemed to reduce productivity, which is, of course, negative, and in contrast to claims in the literature and among leading practitioners. A consequence of these findings for practice is that a team would need to balance WIP depending on the priority of lead time versus productivity. Still, many other factors also affect what is optimal in a given context, so a variety of follow-up studies are needed before any clear advice can be given.

I hope the approach and the challenges discussed in this paper will help other to design similar but improved studies. Follow-up studies could be other case studies but also experiments. It is certainly difficult to conduct such studies without a long series of serious threats to construct validity, internal validity and external validity, but as a community, we may gradually overcome more and more of them.

APPENDIX

Figs. 4a to 4e show the mean WIP, lead time and productivity for each quarter for each of the five teams.

ACKNOWLEDGMENTS

I thank Jørgen Solberg and Anders Johnsen at Software Innovation AS for providing me with development process data, Truls Skeie and Rune Granerud for insightful discussions, Skeie also for calculating WIP values, Lenore Hietkamp for editing the paper, and the anonymous referees for valuable comments.

REFERENCES

[1] H. Kniberg and M. Skarin. 2010. *Kanban and Scrum—making the most of both*. Lulu.com.

[2] D. Anderson, G. Concas, M.I. Lunesu, and M. Marchesi. 2011. Studying Lean-Kanban approach using software process simulation. In *International Conference on Agile Software Development*. Springer, Berlin, Heidelberg, 12-26.

[3] J.D. Little and S.C. Graves. 2008. Little's law. In *Building intuition: insights from basic operations management models and principles* (Vol. 115), D. Chhajed and T.J. Lowe (Eds.). Springer Science & Business Media. Springer US, 81-100.

[4] D.J. Anderson. 2010. *Kanban: successful evolutionary change for your technology business*. Blue Hole Press.

[5] D.J. Anderson, G. Concas, M.I. Lunesu, M. Marchesi, M., and H. Zhang. 2012. A comparative study of Scrum and Kanban approaches on a real case study using simulation. In *International Conference on Agile Software Development*. Springer, Berlin, Heidelberg, 123-137.

[6] G. Concas, M.I. Lunesu, M. Marchesi, and H. Zhang (2013). Simulation of software maintenance process, with and without a work-in-process limit. *Journal of Software: Evolution and Process*, 25(12), 1225-1248.

[7] Al-Baik and J. Miller (2015). The Kanban approach, between agility and leanness: a systematic review. *Empirical Software Engineering*, 20(6), 1861-1897.

[8] M.O. Ahmad, J. Markkula, and M. Oivo. 2013. Kanban in software development: A systematic literature review. In *39th Euromicro Conference on Software Engineering and Advanced Applications (SEAA2013)*. IEEE, 9-16

[9] D.I.K. Sjøberg, A. Johnsen, and J. Solberg (2012). Quantifying the effect of using Kanban versus Scrum: A case study. *IEEE software*, 29(5), 47-53.

[10] D.I.K. Sjøberg, B. Anda, E. Arisholm, T. Dybå, M. Jørgensen, A. Karahasanovic, E. Koren, and M. Vokác. 2002. Conducting realistic experiments in software engineering. In *International Symposium on Empirical Software Engineering (ISESE)*. IEEE, 17-26.

[11] L.J. Cronbach and P.E. Meehl (1955). Construct validity in psychological tests. *Psychological bulletin*, 52(4), 281.

[12] D.I.K. Sjøberg, A. Yamashita, B.C.D. Anda, A. Mockus, and T. Dybå (2013). Quantifying the effect of code smells on maintenance effort. *IEEE Transactions on Software Engineering*, 39(8), 1144-1156.

[13] T. Skeie. 2014. *Does limit on Work-In-Progress (WIP) in software development matter?* MSc thesis. Department of Informatics, University of Oslo, Norway.

[14] G.R. Bergersen, J.E. Hannay, D.I.K. Sjøberg, T. Dybå, and A. Karahasanovic. 2011. Inferring skill from tests of programming performance: Combining time and quality. In *International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE, 305-314.

[15] S. Holm (1979). A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, 65-70.

[16] T.V. Perneger (1998). What's wrong with Bonferroni adjustments. *Bmj*, 316(7139), 1236-1238.

[17] R.A. Armstrong (2014). When to use the Bonferroni correction. *Ophthalmic and Physiological Optics*, 34(5), 502-508.

[18] Y. Lindsjørn, D.I.K. Sjøberg, T. Dingsøyr, G.R. Bergersen, and T. Dybå (2016). Teamwork quality and project success in software development: A survey of agile development teams. *Journal of Systems and Software*, *122*, 274-286.

[19] G.R. Bergersen, D.I.K. Sjøberg, and T. Dybå (2014). Construction and validation of an instrument for measuring programming skill. *IEEE Transactions on Software Engineering*, 40(12), 1163-1184.

[20] V.G. Stray, N.B. Moe, and T. Dingsøyr. 2011. Challenges to teamwork: a multiple case study of two agile teams. In *International Conference on Agile Software Development*. Springer, Berlin, Heidelberg, 146-161.

[21] T. Dybå, D.I.K. Sjøberg, and D.S. Cruzes. 2012. What works for whom, where, when, and why? On the role of context in empirical software engineering. In *International Symposium on Empirical Software Engineering and Measurement (ESEM)*. ACM-IEEE, 19-28.

[22] B.C.D. Anda, D.I.K. Sjøberg, and A. Mockus (2009). Variability and reproducibility in software engineering: A study of four companies that developed the same system. *IEEE Transactions on Software Engineering*, 35(3), 407-429.

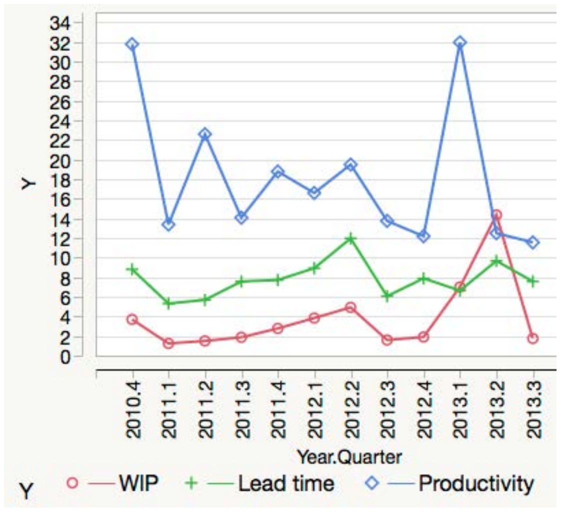

Figure 4a: Team A.

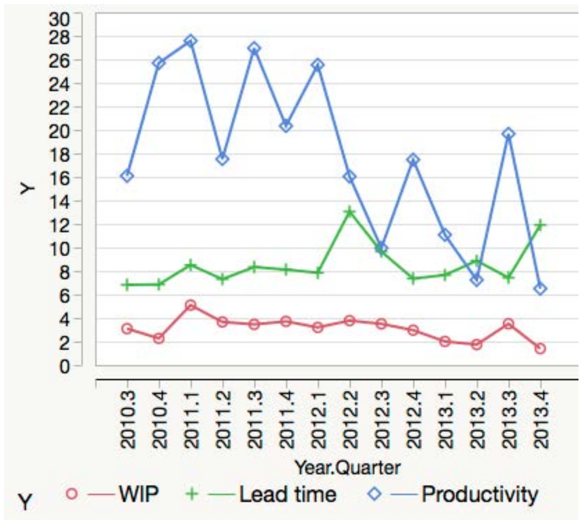

Figure 4b: Team B.

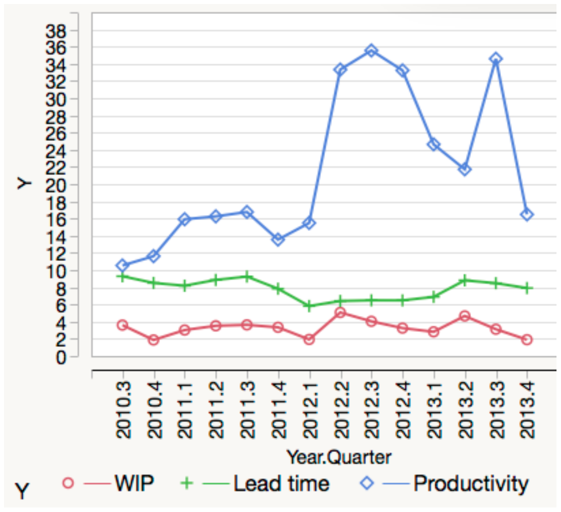

Figure 4c: Team C.

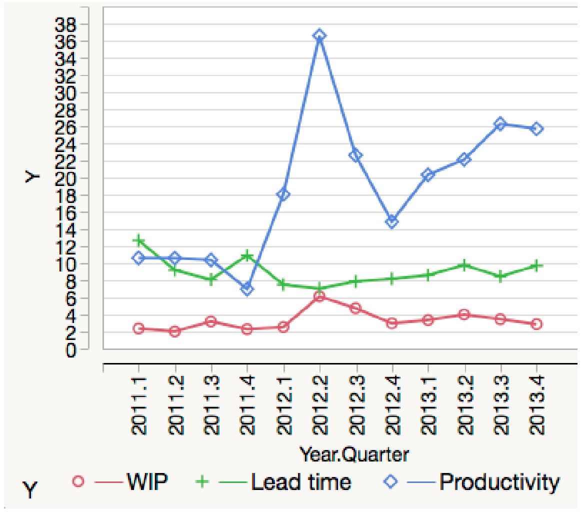

Figure 4d: Team D.

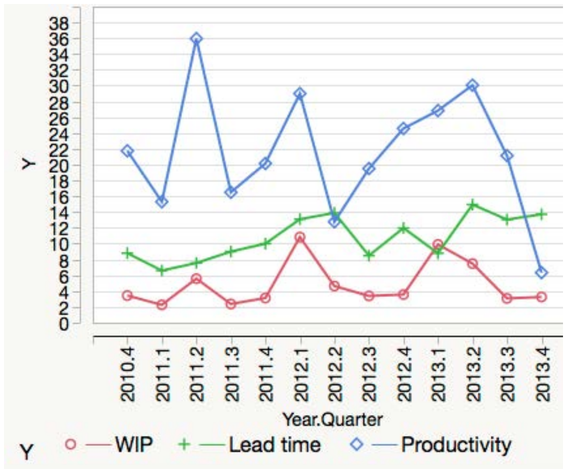

Figure 4e: Team E.