UiO : Department of Informatics
University of Oslo

Do Work in Progress (WIP) - Limit in Agile Software Development Matter?

Truls Skeie

Master's Thesis Autumn 2013

Do Work in Progress (WIP) - Limit in Agile Software Development Matter?

Truls Skeie

18th December 2013

Abstract

Contents

List of Figures

List of Tables

Preface

Part I

Introduction

Chapter 1

Background

In the field of WIP and if WIP matters in software development, lacks proper research, but Giulio Concas and Hongyu Zhang has done research on the difference between limit WIP and unlimited WIP (Concas et al., 2013).

Work in progress is a tool that helps teams limits their tasks by specifying how many tasks a developer can be assigned to at once. WIP helps team to reduce overhead, decrease leadtime and increase throughput

How to find the best WIP in a given interval and context also lacks proper research, but in manufacture business some research has been done. Taho Yanga, Hsin-Pin Fub, Kuang-Yi Yanga stated that WIP could be defined as; WIP = cycle time * throughput rate in manufacture business (Taho Yanga and Yanga, 2007).

1.1 Kanban

"We can define Kanban software process as a WIP limited pull system visualized by the Kanban board" (Anderson et al., 2011).

Kanban system focus on;

- Continuous flow of work
- No fixed iterations or sprints
- Work is delivered when it?s done
- Teams only work on few tasks at the time specified by the limit WIP

- Make constant flow of released tasks

(Anderson et al., 2011).

Toyota production system was introduced Kanban during late 1940s and early 1950s in order to catch up with the American car industry (Ohno, 2001) . In the last ten years software Development Company have started to implement agile methods and Kanban is one of them (Conboy, 2009). Kanban splits one big problem into many small pieces of problems.

When the small pieces are in order, the problems are put up on the Kanban-board to visualize the problems and see potential bottlenecks. When people started to understand Kanban, they easily discovered where the bottlenecks where, and started to help where the bottlenecks where (Shinkle, 2009).

One of the most important people in Kanban software development, David Anderson also referred to as "father of Kanban in the software development industry" (Gupta, 2013), author of Kanban: Successful Evolutionary Change for Your Technology Business once stated "If you think that there was Capability Maturity Model Integration, there was Rational Unified Process, there was Extreme Programming and there was Scrum, Kanban is the next thing in that succession." (Leonardo Campos, 2013)

More and more software projects adapt to Kanban, and this is one of the reasons why this thesis will focuses on Kanban and WIP. Kanban is one of the agile method in the wind these days, and is used with Lean Software development which is one of the fastest growing approaches in software development (Anderson et al., 2011)

One of the main difference between Scrum and Kanban is estimation, in simulation of software maintenance process, with and without a work-in-process limit (Concas et al., 2013) estimation was defined to be the main source of waste. In their research, they find out, if they let the developers work with small tasks at time and not be interrupted, they will be more effective. The developers in this case was interrupted when they was assigned to estimate tasks. The research groups decided to implement Lean-Kanban, which includes minimizing waste, which meant estimation for this case. After implementing Lean-Kanban the teams increased the ability to perform work, lower the lead time and meet the production dates.

1.2 Scrum

Scrum has three main roles, the Product Owner, the Scrum Master and the members of the development team. The Product owner in collaboration with the Scrum Master decides which work to be prioritized in the backlog. The Scrum Master acts like a team leader and helps the team and organization to take best advantages of Scrum. The development team works on tasks specific for the iteration there in (Alliance, 2012).

Iteration is a time-boxed interval over a given time, usually from one to four weeks. Before each sprint, a sprint planning meeting is conducted, with all the team members attending (ibid.).

In each sprint a minimal backlog is created so the developer knows which tasks to work on. The Product Owner and the team members discuss and decide which tasks from the backlog to be added to the minimal backlog (ibid.).

After the minimal backlog is full, the Product Owner and the team members discuss each task from in order to get a better understanding and a shared understanding of what is required in order to complete the task (ibid.).

Scrum requires that a new feature is ready for release after a sprint. The feature should be a visible part of the product in order to get feedback from end-users (ibid.)

1.3 Lean-Kanban

The Lean approach was introduced between 1948 and 1975 in manufacturing work in Japan. It was designed to find and eliminate waste, so the manufacturing could deliver value to the costumer more efficiently. In 2003 Mary and Tom Poppendieck first introduced Lean thinking to software development, they published ?Principles of Lean Thinking? (M. Poppendieck and T. Poppendieck, 2003). An important tool to manage work flow is the concept of pull-systems, which means tasks are put in production only when a costumer asks for it (M. Poppendieck and T. Poppendieck, 2009). In the last years, Kanban has been introduced more and more to software development, and is becoming one of the keys to Lean practice in the field (Anderson et al., 2011).

1.4 WIP

"There's stated when using short-cycle times and Kanban board to limit WIP, the software development team's learning is increased" (Middleton and Joyce, 2012)

Work in progress (WIP) is a tool in Kanban to reduce overhead by limit task-switching for each developer and visualize bottlenecks. Each column in a Kanban board usually has a specific WIP as illustrated in figure 1.1.

WIP can be explained using cars and roads as analogy. All roads have it maximum capacity width number of cars. When this limit is reached, traffic jam occurs and the throughput of cars decreases and lead time increases. The same can be said about software development teams, a software team has a maximum number of tasks they can perform, if the team is pushed over the maximum limit, the throughput of tasks decreases and lead time increases.

But in order to get the best throughput and lead time, a software team needs to experiment with WIP. When first implementing Kanban, Shinkle explains that the users don't care about the WIP, but rather the visibility of Kanban. When the user gets more experience with Kanban, they start to attempt limit WIP (Shinkle, 2009) Shinkle defined novice and more experience kanban-users with a descriptive analogy (ibid.): "Think about a typical person wanting to bake a cake. They go to the store, purchase a boxed cake mix, and follow the directions as described on the back of the box. They have little to no knowledge about how to alter the recipe nor do they have a desire to do so. Their goal is simply to bake a cake." "An advanced beginner understands how to apply some context to the instructions or rules on the back of the box. They can make minor adjustments for things like altitude, pan size, oven conditions, etc. They are still following the basic recipe, but can make minor adjustments likely based on previous experiences." In the paper Applying the Dreyfus Model of Skill Acquisition to the Adoption of Kanban Systems at Software Engineering (SEP) (ibid.), Shinkle interview some developers and one developer said: ?WIP limits seem to be the worst understood part of the Kanban system. When used properly, it exposes bottlenecks and reduces lead time for individual work items. Used improperly, it can starve developers for work or result in too many people working on the same work items." (ibid.)

Setting a limit on WIP has a number of benefits according to Hopp, Spearman and Suri. They stated that it reduces flow times, reduces variation and improves quality. However Srinivasan, Ebbing and Swearing said that setting the WIP is not easy. They suggest that WIP are just set, and then observe throughput, and adjust after that (Srinivasan, Ebbing and Swearingen, 2003). Several people have their opinion on how to best

measure the WIP, but most of them agree upon that's there is no clear rule of how to measure WIP in software development. Lukasz proposes to use The Effectiveness as an indicator measured at the end of each cycle. The effectiveness is a formula that takes the number of bugs found (ai) and the number of bugs found by external people (e.g. lawyers, accountants, coaches, consultants, translators, internal and external service providers ,etc.)(ei), and minus ai and ei, then divide the result by ai and multiply it by 100% as shown in 1.1 (Sienkiewicz, 2012)

$$Ei = \frac{ai - ei}{ai} * 100\% \tag{1.1}$$

On the other hand, Kanban says you should limit WIP. So what should the limit be? - Kanban; Don't know, experiment (Kniberg, 2010).

Lean Software Management suggest that WIP should be minimized, to keep high quality (Ikonen et al., 2011) and to create continuous flow and bring problems to the surface (Middleton and Joyce, 2012). In Lean Software Management: BBC Worldwide Case Study when the team realized the where bottlenecks and to high WIP the team started to determine WIP by their constraints. The team quickly realized that they had fewer quality assurance/testing staff and business analysts than software developers. This reflected the bottlenecks, so the team adjusted WIP to how much work they could handle; this gave the team more experience in dealing with WIP (ibid.).

In the authors of Lean thinking in software development: Impacts of kanban on projects suggest that task should be prioritized from high to low, in order to keep the best WIP. Lean thinking in software development: Impacts of Kanban on projects stated in the conclusion that high WIP will keep people task switching and not be able to fully concentrate on each work in progress (Ikonen et al., 2011). As notice, there?s no clear rule on how to deal with WIP and how to determine WIP, although WIP is a crucial tool in order to Kanban sufficient

1.5 Kanban Board

"The Kanban board makes it clear to all the team members the exact status of progress, blockages, bottlenecks and they also signal possible future issues to prepare for"(Middleton and Joyce, 2012).

The Kanban board is one of the most important tools in Kanban. It?s used to minimize WIP, increase the information flow, with visualization and spot bottlenecks (Concas et al., 2013). The Kanban board uses named columns

to illustrate where each item is in the workflow. Every column has a WIP, to specify how many works in progress there are allowed in the column (Middleton and Joyce, 2012). A Kanban board is illustrated in figure 1.1, where each column has a name respectively Backlog, In progress and done.

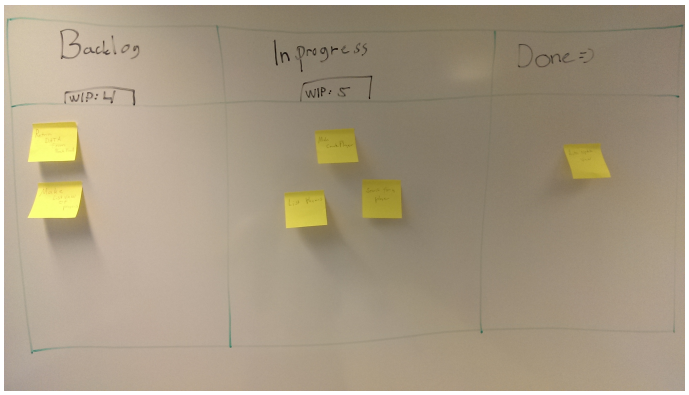

Figure 1.1: Example of a Kanban board

1.6 Lead time

"Lead time is the total elapsed time from when a customer requests software to when the finished software is released to the custome" (ibid.). The citation by Joyce above is the close to definition of what lead time is. This definition could be useful for consultancy companies, but for SI, who is a big in-house development company with few releases each year this definition is unsuitable. Quantifying the effect of Using Kanban versus Scrum stated to definition of why the definition is unsuitable:

"First, the amount of time a work item remains in the backlog queue before it's 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 might propose new features themselves and put them on the backlog before any customers request them. Second, given a policy of two or three releases a year, the result of a work item isn't delivered to the customer immediately after it's finished" . (Sjøberg, Johnsen and Solberg, 2012)

As long as SI has defined what Lead time is in their case, lead time could be used as an essential ingredient when you look for the optimal WIP for their project, because with lead-time, they can control the timer interval for each task. Often in project, lead time is split into pieces, so every task has its own lead-time; this gives the development teams the advantages to experiment with different WIP's in order to see the different lead-times and

then measure which WIP that suits this project the best.

1.7 Throughput

”The average output of a production process (machine, workstation, line plant) per unit time (e.g., parts per hour) is defined as the systems throughput or sometimes throughput rate” (Adams and Smoak, 1990)

The concept of throughput in order to measure how productive teams, people or companies is similar in both software development and manufacturing. But, in software development each tasks is abstract, tasks can have different solutions depending on how the team or developer approach and the tasks is usually only done once.

In manufacturing, each task usually has one solution and when the solutions are found the physical item is mass produced. Adams said "Throughput in plant, line or workstation, is defined as the average quantity of good parts produced per unit time" (ibid.), which gives a good example of the relationship between tasks in manufacturing and software development. In manufacturing the part either fits its purpose (good) or not (defective), in software development a tasks can fit a purpose, but the purpose may be wrong. As long as the software development task is bug free, it’s delivered as non-defective but it may not fit the defined purpose by the end users.

Throughput is measured in number of finished delivered tasks per day, week, month, quarter or year. In order to make throughput measurement successful, each tasks has to require almost the same amount of work.

For instance; team x had a throughput of ten after the first quarter, twenty after the second, fifteen after the third and twelve after the last quarter and they used Scrum the first two quarters and Kanban the last two as illustrated in table 1.1 It will look like team x benefits most from Scrum. But if the task during the Kanban time was twice the size of Scrum, the Kanban approach would fit them better than Scrum. In order to find that out, the team x needs to standardize how much work each tasks should require.

Quarter	Throughput	Framework
1	10	Scrum
2	20	Scrum
3	15	Kanban
4	12	Kanban

Table 1.1: Throughput

1.8 Chrun

1.9 Limit WIP vs. Unlimited WIP

There's been done some research on how the throughput, leadtime and how developers experience WIP limit and unlimited WIP

Simulation of software maintenance process, with and without a work-in-process limit has done research on limit WIP vs. unlimited WIP, one of the result from this paper was at the end of a simulation, the average of closed issued was 4145 when the WIP was limited and 3853 when the limit was not limited (about 7% less). The paper concludes their finds like; developers are more focused on fixing few issues, because the number of issues they can work on is limited. Because the limit of WIP, the developers are more likely to continue on the issue from the day before, rather than starting on another issue, this reduce overhead. When developers start on a new issue, they need to use time to familiarize themselves with the code and the issue. That could create unnecessary overhead if some developer already has done it, but that developer is now working on a another issue (Concas et al., 2013).

1.10 Software Innovation(SI)

Software Innovation is a Scandinavian software company. SI develops and delivers market-leading Enterprise Content Management applications that helps organizations improve and increase efficiency in document management, case handling and technical document control.

Software Innovation has approximately 300 employees, and has offices in Oslo, Copenhagen and Stockholm and a development center in Bangalore (*Software Innovation* 2013).

Part II

The project

Chapter 2

Planning the project

Part III

Conclusion

Chapter 3

Results

Bibliography

Adams, M. and B. Smoak (1990). 'Managing manufacturing improvement using computer integrated manufacturing methods'. In: *Semiconductor Manufacturing Science Symposium, 1990. ISMSS 1990., IEEE/SEMI International*, pp. 9–13. DOI: 10.1109/ISMSS.1990.66111.

Alliance, Scrum (2012). 'Scrum, A description'. In:

Anderson, David et al. (2011). 'Studying Lean-Kanban Approach Using Software Process Simulation'. In: *Agile Processes in Software Engineering and Extreme Programming*. Ed. by Alberto Sillitti et al. Vol. 77. Lecture Notes in Business Information Processing. Springer Berlin Heidelberg, pp. 12–26. ISBN: 978-3-642-20676-4. DOI: 10.1007/978-3-642-20677-1_2.

Conboy, Kieran (2009). 'Agility from First Principles: Reconstructing the Concept of Agility in Information Systems Development'. In: *Information Systems Research* 20.3, pp. 329–354. DOI: 10.1287/isre.1090.0236.

Concas, Giulio et al. (2013). 'Simulation of software maintenance process, with and without a work-in-process limit'. In: *Journal of Software: Evolution and Process* 25.12, pp. 1225–1248. ISSN: 2047-7481. DOI: 10.1002/smr.1599.

Gupta, Vikram (May 2013). *InfoQ Interviews David J. Anderson at Lean Kanban 2013 Conference*. URL: http://www.infoq.com/articles/David_Anderson_Lean_Kanban_2013_Conference_Intervew (visited on 01/10/2013).

Ikonen, M. et al. (2011). 'On the Impact of Kanban on Software Project Work: An Empirical Case Study Investigation'. In: *Engineering of Complex Computer Systems (ICECCS), 2011 16th IEEE International Conference on*, pp. 305–314. DOI: 10.1109/ICECCS.2011.37.

Kniberg, Henrik (2010). *Kanban and Scrum - making the most of both*. lulu.com. ISBN: 0557138329.

Leonardo Campos Rafael Buzon, Eric Fer (Mar. 2013). *Kanban Pioneer: Interview with David J. Anderson*. URL: http://www.infoq.com/articles/David-Anderson-Kanban/ (visited on 30/09/2013).

Middleton, P. and D. Joyce (2012). 'Lean Software Management: BBC Worldwide Case Study'. In: *Engineering Management, IEEE Transactions on* 59.1, pp. 20–32. ISSN: 0018-9391. DOI: 10.1109/TEM.2010.2081675.

Ohno, Taiichi (2001). *Toyota Production System on Compact Disc: Beyond Large-Scale Production*. Productivity Press. ISBN: 1563272679.

Poppendieck, Mary and Tom Poppendieck (2003). *Lean Software Development: An Agile Toolkit*. Addison-Wesley Professional. ISBN: 0321150783.

– (2009). *Leading Lean Software Development: Results Are not the Point*. Addison-Wesley Professional. ISBN: 0321620704.

Shinkle, C.M. (2009). 'Applying the Dreyfus Model of Skill Acquisition to the Adoption of Kanban Systems at Software Engineering Professionals (SEP)'. In: *Agile Conference, 2009. AGILE '09.* Pp. 186–191. DOI: 10.1109/AGILE.2009.25.

Sienkiewicz, Lukasz (2012). 'Scrumban - the Kanban as an addition to Scrum software development method in a Network Organization'. In:

Sjøberg, D.I.K., A. Johnsen and J. Solberg (2012). 'Quantifying the Effect of Using Kanban versus Scrum: A Case Study'. In: *Software, IEEE* 29.5, pp. 47–53. ISSN: 0740-7459. DOI: 10.1109/MS.2012.110.

Software Innovation (Dec. 2013). URL: http://www.software-innovation.com/EN/COMPANY/pages/default_.aspx (visited on 12/12/2013).

Srinivasan, Mandyam M., Steven J. Ebbing and Alan T. Swearingen (2003). 'Woodward Aircraft Engine Systems Sets Work-in-Process Levels for High-Variety, Low-Volume Products'. In: *Interfaces* 33.4, pp. 61–69. DOI: 10.1287/inte.33.4.61.16377.

Taho Yanga, Hsin-Pin Fub and Kuang-Yi Yanga (2007). 'An evolutionary-simulation approach for the optimization of multi-constant work-in-process strategy'. In: 107, pp. 104–114.