To be appeared: Ikonen, M. & Kettunen, P. & Nilay, O. & Abrahamsson, P. in Proc. of SEAA Euromicro 2010 (IEEE).

Exploring the Sources of Waste in Kanban Software Development Projects

Marko Ikonen[1] Petri Kettunen[1] Nilay Oza[2] Pekka Abrahamsson[1]

[1)]{marko.ikonen|petri.kettunen|pekka.abrahamsson}@cs.helsinki.fi,
Department of Computer Science, University of Helsinki, Finland
[2)]nilay.oza@vtt.fi, VTT Technical Research Centre of Finland

Abstract

The application of agile software methods and more recently the integration of Lean practices contribute to the trend of continuous improvement in the software industry. One such area warranting proper empirical evidence is a project's operational efficiency when using the Kanban method. This short paper takes a new angle and explores waste in the Kanban-driven software development project context. A preliminary research model is presented for helping the consequent replication of the study. The results from the empirical analysis suggest Kanban can be an effective method in visualizing and organizing the current work, but does not prevent waste from creeping in, although the overall project outcome may be successful.

1 Introduction

The software industry is in constant search of new ways of solving existing problems [6]. The goals vary in different improvement initiatives ranging from resolving time-to-market delay to reducing operation costs and increasing productivity. One of the newest fashions in the software industry is the attempt to apply lean production principles to software development. One of the key traits of lean production principles is eliminating all kinds of waste from the development. Consequently, similar ideas for waste removal have been proposed to be adopted for software product development [12]. One of the lean tools is the Kanban way of managing production operations [9]. Kanban has been applied to software production as a project management process model [3]. The buzz is already in motion on combining Kanban with the well established Scrum method in Agile software development. E.g. Kniberg [8] introduces similarities and differences between Kanban and Scrum. Yet, to our knowledge there are few, if any, empirical studies exploring the internal dynamics or the process impact of the Kanban model in software engineering. Therefore, following this line of *lean* thinking leads to one of the research problems: seeing waste in Kanban-driven software projects.

This paper explores different sources of waste in a case study of such a software development. We employed a controlled case study research approach [14] in a novel experimental software R&D laboratory called Software Factory [1] wherein the empirical evaluation was performed based on a 7-week,13-person software development project producing a real web service for tens of thousands of users. The results reveal that while Kanban has been designed to minimize non-value-adding activities, there are still several sources of waste that are likely to appear. In addition, the preliminary research model proved to be proficient for identifying potential sources of waste although it requires further development. Altogether this research endeavor suggests practical as well as methodological implications.

2 Related Work

In lean thinking, *waste* is basically everything that does not add to the customer value of the products. In general, three basic categories of waste-related elements can be identified: (1) Muda: non-value-adding activities (NVA), (2) Mura: variations (in process quality, cost, delivery), and (3) Muri: unreasonableness (overburden). Consequently, one of the most important principles of lean software development is to learn to recognize waste. In software development, such elements can be interpreted as follows [12]: partially done work (inventory); extra processes (NVA); extra features (overproduction); task switching, waiting, motion (NVA); and defects. Table 1 illustrates why these issues are considered waste. Furthermore, some less obvious waste in certain cases can be wasted development time (e.g. due to fuzzy front-end activities), wasted progress (requirements not tied to business priorities), and wasted intelligence (underutilization of personnel competence) [7].

In manufacturing operations, the different kinds of waste are basically straightforward to detect by observing the physical material flows and machine/worker activities [2,

Element	Rationale for considering as waste
partially done work	• does it work and really solve the business problem after integrated into the software environment? • ties up resources
extra processes	• unnecessary paperwork consumes resources and adds no value for the customer
extra features	• consumes resources when an extra feature of a system is tracked, compiled, integrated, and tested • potential failure point
task switching	• working in multiple teams causes more interruptions • re-orientation back to work takes time
waiting	• delays in starting a project, staffing, reviews, approvals, testing, etc. add no value • prevents realizing value for the customer as fast as possible
motion	• lack of immediate access to other developers and appropriate representatives disrupts concentration, and reestablish focus takes time • regarding artifacts (e.g. documents or code), tacit knowledge does not move with the handoffs from the next person in line
defects	• a minor defect discovered after weeks is more time-consuming than a major defect detected in a minute

Table 1. Reasoning for considering the seven elements as waste [12].

10]. Invisibility of waste in software development, however, restrains the progress. Partially done work, for example, involves such emerging issues like requirement and error inventories. In order to manage these inventories, extra processes are typically generated, which creates waste.

The Kanban process model is one of the key operation management tools in lean manufacturing [9]. It is basically a flow control mechanism for pull-driven Just-In-Time production in which the upstream processing activities are triggered by the downstream process demand signals. In general, Kanban has three rules: (1) visualize the workflow, (2) limit work in progress (WIP) at each workflow state, and (3) measure the lead time (i.e. average time to complete one item) [8]. Advantages of the Kanban-driven operations are that the inventories (simultaneous WIP) are reduced and the overall production flow can be balanced easier.

Poppendieck and Poppendieck [12] suggest 22 "thinking" tools for practitioners of lean software development and categorize them into seven principles. Following their suggestion of *seeing waste* being the most important tool to begin with, our research here focuses on revealing waste in Kanban software development projects. Figure 1 presents this focus. We applied that tool framework as the research model (Section 4.1).

3 Study Design

The *Software Factory* is a new software engineering research and education setting at the University of Helsinki [1]. It is an advanced R&D laboratory environment for conducting software projects. The concept comprises the physical laboratory environment coupled with a unique operational model from the empirical research perspective: the operational setting comprises three different usage perspectives of the laboratory: business (entrepreneurship), education, and research.

The case project is a Kanban-based software development project. The project size was a thirteen-member software team, and an external technical consultant who visited the Software Factory daily. The participants were Master's students with varying amounts of work experience in industry. The project lasted seven weeks and the participants worked six hours per working day, which resulted in 23 person months of effort investment during the project. The members were accepted to the project according to their interests, and the Bachelor degree was required. In addition to technical knowledge (i.e. at least coding and testing skills), each member in the team had some experience of working in a team-based environment. Some of the team members were also experienced in project management.

Besides the use of Kanban, no other particular development method was insisted upon. The team had close to full control within the R&D setting to decide upon the practices used. As a result, coding and testing were executed mostly individually after the first two weeks. More experienced members worked in small groups when they made designs. They also helped and assisted the others in technical and practical issues and gave useful advice. In short, the team was self-organized.

Our research aim was to find out what kind of sources of waste can be found in Kanban-based software development projects. For this aim, we developed a preliminary research model (Section 2) designed to identify these sources. We applied the framework illustrated in Figure 1 by generating questions regarding the waste issues under the seven element categories (from A to G). The semi-structured interview method [11] was used for collecting data. Five persons from the team of 13 for the interviews were selected so that they represented the three different roles: less experienced, more experienced, and the team leader. This is called a role-based sampling. The first author performed the semi-structured interviews after the project end. We used open-ended questions and a semi-structured theme inter-

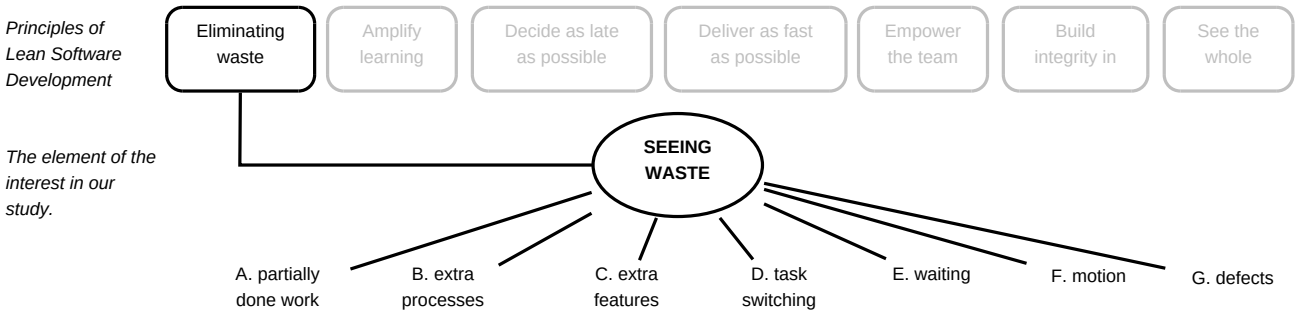

Figure 1. Our preliminary research model based on [12].

view technique. Each interview lasted approximately one hour. The interviews were recorded in audio and the answers were categorized into the seven kinds of waste based on the research model presented in Section 2. In addition to this evaluation, we measured the overall project success based on Shenhar's [15, 16] first (i.e. project efficiency), second (impact on the customer), third (i.e. business success) and fourth (preparing for the future) dimension.

4 Empirical Analysis

The interviews revealed waste and increased understanding of the functioning of the Kanban process model. We organized our interview results according to the research model into seven categories of waste (from A to G), based on Figure 1. Section 4.1 assembles the interviews by these categories with sample questions and responses, and explores waste found (see Table 2) in the project while Section 4.2 determines the overall success of the project.

4.1 Waste Found in the Project

4.1.1 Waste A: Partially done work

- *Did anyone have unfinished tasks before a new task assignment?*
- *What were the reasons for this simultaneous task assignments?*

A team member answered: *"Actually, there was no major partially done work because the Kanban board allowed only two tasks in each column per developer. Mostly, people did a single task from the beginning to the end before they were assigned to a new task."*

Partially done work was mostly present in an individual sense, when members started tasks but then they had to reassign the tasks to more skillful persons who finished the tasks. Another member commented: *"We had to reassign some tasks to more skillful persons because of inexperience and lack of routines in some junior members."*

However, only one major problem in this waste category was found when another member stated: *"One task was estimated to be ready within half an hour. However, it took over two days, which blocked a couple of other tasks until the delayed task was accomplished."*

Sometimes, members had to implement two tasks at the same time. A member explained the reason for this: *"Sometimes, when a person was assigned to a task, it turned out that it cannot be done because it needs some other task that has not been done yet."*

Partially done work showed several indicators of waste within the project operation. It emerged from different types of inadequacies at different levels in the system. This form of waste also generated redundant task switching efforts.

4.1.2 Waste B: Extra processes

- *What unnecessary processes were there in the project in your opinion?*
- *What kinds of processes were eliminated during the project?*

One member described some unnecessary processes: *"We discussed in our retrospectives that all code review and quality assurance actions would not have been needed for small tasks [these two actions were accepted as columns in the Kanban board so these actions had to be done]. Another inefficient process we agreed was retrospectives at the beginning of the project: they took almost all day for the whole team to process and analyze what was bad, what was good, what had been done and what will be done."*

By asking about extra processes, the questions revealed such processes but also the team's capability to adapt to changes by optimizing their processes.

4.1.3 Waste C: Extra features

- *Did you design or implement any extra features that the customer did not ask for?*
- *If any, what were the reasons for that? How relevant were they in the customer's opinion?*

A member stated: *"Some extra features were produced without consulting the customer. This was a result from the customer representatives' unawareness of what they really needed. So we thought these extra features are important. There were a few big extra elements and some smaller ones. When we then demonstrated these features, the customer praised our insight."* and then specified these extra features: *"They were not 'extra hyping' but intuitive increments for usability of the interface, for example."*

Another member commented: *"Without comments from the customer, we usually implemented only such extra features we were quite sure that the customer wants. These implementations related mostly to certain basic features."*

Some extra features were implemented in the project particularly to increase usability. While extra features increase the possibility of defects and malfunctioning and take time, they actually benefitted the customer of this project.

4.1.4 Waste D: Task switching

- *If there were any overlapping task assignments for one person, how difficult was it to reorient different tasks simultaneously?*
- *How much did people have to suspend their own tasks in order to help others? How badly did this disturb their own work? E.g. did the quality decrease?*

A member answered: *"Primarily, people carried out one task at a time. If two tasks needed each other in order to be completed, then a person might carry out two tasks simultaneously. Another exception was the code review [a column in the Kanban board, which required someone else to review this code]: if no one was currently available to make the code review, then the person might be assigned to a new task... If a task required contribution from more than one person, we attempted to break it into smaller parts."*

Another viewpoint regarding the task switching was revealed in a member's answer: *"When I help someone, it usually takes time to orientate to the task already started. When I do tasks assigned to me and then someone asks for help, it does not take so long to re-orientate to my tasks again, except when I have just started a new task."*

As the project demonstrated, task switching can be used as an option for waiting. On the other hand, task switching was time-consuming and created some waste.

4.1.5 Waste E: Waiting

- *What things or people did you have to wait for (e.g. customer-related or technical support)?*
- *What were the issues you had to wait for most of all and how long did it take?*

One member had noted the following on waiting: *"Some tasks ended up in the code review column [of the Kanban board] for waiting for a free reviewer for a long time. The implementers, however, spent this waiting time efficiently by carrying out other tasks or by helping other people... When we needed IT support for temporal hardware or software problems, the waiting time was no more than one day."*

Another member stated: *"Sometimes I needed more information from that senior member who had designed task tickets. If he was occupied, I had to wait since I didn't want to put effort into implementing something in the wrong way... Another thing we had to wait for was delayed customer demos. Fortunately, these delays happened seldom and were about 15 minutes only. The only big deal was the headsets which were ordered at the beginning but they arrived at the end of the project. For this reason, we could not watch important screen casts because the audio would have disturbed other people's work."*

Another member commented that: *"Because the tasks were partitioned into small pieces, excluding a couple of exceptions, people mainly did not have to wait for the accomplishment of other tasks."*

Waiting is an inseparable part of projects. While some waiting can be tolerated, damages for a project caused by waiting must be eliminated to have more efficient progress.

4.1.6 Waste F: Motion

- *How much did people have to move around physically inside or outside the workspace and how much did it disturb your progress or concentration?*
- *What problems occurred with sharing information?*

Because of the ideal work space for the project, we did not find any relevant motion waste in physical terms. A member commented: *"My only visit was the IT support unit [located on the same floor]. Some members moved around much more than at lunch breaks but these related to getting some extra snacks, for example, not to the process. This moving around didn't disturb me."*

An environment with well-designed hardware, appropriate software tools, and services makes motion greatly unnecessary since there is no need to visit outside the workspace in order to get things done. Problems with sharing information, however, caused duplicate work as another member admitted: *"It happened (once) that two members were carrying out the same task by accident."*

4.1.7 Waste G: Defects

- *How fast did you detect bugs or malfunctioning?*
- *How fast did you react to these detections?*

The attempt was made to optimize detection time in order to minimize the causes of the defects: *"Bugs were fixed immediately after noticing them... We used much TDD and BDD in order to find malfunctioning. Despite this, testing during the last week revealed some serious issues."* In other words, defects noticed at last-minute encumbered the team.

Waste element	Sources discovered in the case project
A. partially done work	• delays in the critical path • to implement a task required another, yet unimplemented task
B. extra processes	• some Kanban-based routines unnecessary for tiny tasks • inefficient retrospective meetings
C. extra features	• unawareness of the key requirements
D. task switching	• two tasks need each other in the implementation phase • disruptions: helping other people or contributing other tasks • avoidance of waiting: utilizing the waiting time by doing other tasks
E. waiting	• for customer representatives • for IT support • for hardware shipments for the project • a person occupied when someone needs assistance
F. motion	• lack of communication
G. defects	• defects found at last-minute

Table 2. Waste and its sources found.

4.2 Degree of Project Success

Regarding Shenhar's first dimension (project efficiency), the project was a success: the limits were not exceeded. The delivery date was fixed and the required functionality changed somewhat during the project. Likewise, the project completed each of the six goals of the second dimension (impact on the customer). In contrast with the customers' expectation, the project exceeded them. Regarding the third dimension (business success), the project met its set target which was to generate interest in the corporate funding schemes. In the terms of the fourth dimension (preparing for the future), the project documented its work and findings for the future Software Factory teams' use. The project outcome is successfully continuing to be developed beyond the alpha-release phase.

5 Discussion

5.1 Findings

Two principal findings can be presented. First, all potential sources of waste proposed in the research model were found in practice at varying levels as summarized in Table 2. Second, finding waste, however, did not significantly contribute to explaining the success of the project.

The level of success was evaluated systematically since it is necessary to relate this to the sources of waste. Hypothetically, it can be argued that the more waste a team produces, the less chances for success it has. The good success of the project is a probable reason explaining why only a small amount of waste was found. Whether this was the direct result of the application of Kanban, experience of the team leader and developers or customer pressure remains harder to pinpoint. Yet, it is argued that waste represents elements that restrain the progress of projects, which endangers their chances for success [5].

The study demonstrates that zero waste is not a requirement for a successful project. When observing the waste and its sources in Table 2, it can be concluded that they are quite minimal. As an example, a team member expressed his distress with having to wait for a customer demonstration. Yet, the extra waiting time was only 15 minutes in this particular case or cases. However, on another occasion, the team had to wait for the headsets for the whole project, which prevented them from using web casts for learning or other purposes. This is a significantly more serious issue. In a large organization, the waiting time may amount to days or even, in some cases, weeks.

Thus, when using the seven possible elements of waste as the lenses for an analysis, it is likely that most of them are apparent in any software development project. What makes a difference is the attempt to minimize their impact or existence. This is where Kanban is likely to be useful. Even if Kanban strives for minimizing the non-value-adding work, waste may very well still creep in as the case study shows. Thereby, by applying Kanban as a means to organize the development, attention must particularly be paid to the following sources of waste: delays in the critical path (waste element A in Table 2), optimizing Kanban board functioning for both tiny and large tasks (B), elimination of inefficient methods (B), more informative customer co-operation in order to diminish unawareness of the customer's needs (C), and clarification of sharing information (F). The team can affect these five sources of waste easier than e.g. limited project resources, such as one person cannot assist many people simultaneously.

Moreover, the results show a dependency between task switching (D) and waiting (E): Members had to wait when they were unable to proceed with their tasks. In order to

spend this waiting time efficiently, they were assigned to other tasks, which resulted in task switching. According to the results, this dependency seems to be only a one-way implication from waiting to task switching, not vice versa. It is proposed that task switching is less time-consuming than waiting whenever it can replace the waste of waiting.

5.2 Limitations of the Study

A qualitative research involves subjectivity of researchers' interpretations and interviewees' limited viewpoints, and it is important to become aware of this problem [18]. One limitation for the study, as often in other qualitative case studies, is the generalisability of the results. However, the aim of this study is to offer first-hand empirical evidence of the application of Kanban for software development from the viewpoint of waste. As a result, it is claimed that while different kinds of waste are found, it is not instrumental in influencing the outcome of the project. This opens up research avenues and hypotheses that can be further studied in other settings. Another limitation to the validity of the proposed findings is the use of students in this study as study subjects. However, it is well established that when one seeks to establish a trend, the use of students is acceptable [17]. Tichy [17] uses a method comparison as a specific target of study where the use of students is a valid approach. Others have made similar suggestions. Höst et al. [4] concluded that students are relevant when considering experimentation in software engineering.

6 Conclusions

By revealing waste in projects, significant actionable opportunities in terms of saving resources and accelerating lead-time can be reached for practical use. According to the results of this study, sources of waste can be tracked in Kanban-based software projects based on the research model (Figure 1). Moreover, the findings can be compared with further studies explored in less successful projects.

The relation between Kanban and waste is an area of study that has not received much attention. An interesting viewpoint is to develop understanding of how seeing and eliminating waste affects this relation. Regarding the focal study, more specific knowledge would be needed in order to determine typical waste in Kanban-driven software projects. Another point of interest is methodologies or tools to recognize this waste. The waste should be conceptualized deeper and connected to other elements of the lean thinking such as flow [13]. In order to make the progress of projects more efficient, 21 other lean "thinking" tools are offered [12]. That opens up new avenues for further research on efficiency improvements also in Kanban-driven projects and expands research to the scope of Lean Enterprise.

7 Acknowledgments

This work was supported by TEKES as a part of the Cloud Software program of Tivit (Finnish Strategic Centre for Science, Technology and Innovation in the field of ICT). The authors also express their gratitude to Software Factory's team #1 and to the reviewers for comments.

References

[1] P. Abrahamsson. Unique infrastructure investment: Introducing the Software Factory concept. *Software Factory Magazine*, 1(1):2–3, 2010.

[2] R. Goodson. Read a plant – fast. *Harvard Business Review*, May 2002.

[3] K. Hiranabe. Kanban applied to software development: From agile to lean, 2008. http://www.infoq.com/articles/hiranabe-lean-agile-kanban.

[4] M. Höst, B. Regnell, and C. Wohlin. Using students as subjects – a comparative study of students and professionals in lead-time impact assessments. *Journal of Empirical Software Engineering*, 5(3):201–214, 2000.

[5] M. Ikonen and P. Abrahamsson. Anticipating success of a business-critical software project: A comparative case study of waterfall and agile approaches. In *ICSOB '10*. Springer-Verlag, 2010.

[6] I. Jacobson. What they dont teach you about software at school: Be smart! In *XP 2009*, pages 1–4. Springer, 2009.

[7] N. Kindler, V. Krishnakanthan, and R. Tinaikar. Applying lean to application development and maintenance. *McKinsey Quarterly*, (Spring), 2007.

[8] H. Kniberg. Kanban vs. scrum – how to make the most of both, 2009. http://www.crisp.se/henrik.kniberg/Kanban-vs-Scrum.pdf [18-Jan-2010].

[9] J. Liker. *The Toyota Way*. McGraw-Hill, USA, 2004.

[10] K. Lui and K. Chan. *Software Development Rhythms*. John Wiley & Sons, New Jersey, USA, 2008.

[11] M. Q. Patton. *Qualitative evaluation and research methods*. Sage, 2 edition, 1990.

[12] M. Poppendieck and T. Poppendieck. *Lean software development: An agile toolkit*. Addison Wesley, Boston, Massachusetts, USA, 2003.

[13] D. G. Reinertsen. *The Principles of Product Development Flow: Second Generation Lean Product Development*. Celeritas Publishing, Redondo Beach, CA, USA, 2009.

[14] O. Salo and P. Abrahamsson. Empirical evaluation of agile software development: The controlled case study approach. In *PROFES '04*, pages 408–423. Springer, 2004.

[15] A. J. Shenhar, D. Dvir, O. Levy, and A. C. Maltz. Project success: A multidimensional strategic concept. *Long Range Planning*, 34:699–725, 2001.

[16] A. J. Shenhar, O. Levy, and D. Dvir. Mapping the dimensions of project success. *Project Management Journal*, 28(2):5–13, 1997.

[17] W. Tichy. Hints for reviewing empirical work in software engineering. *Journal of Empirical Software Engineering*, 5(4):309–312, 2000.

[18] R. Yin. *Case Study Research: Design and Methods*. Sage: Thousand Oaks, CA, USA, 1994.