

Aalto University
School of Science
Degree Programme in Information Networks

Eevert Saukkokoski

What exactly is the use of dailies:

Knowledge co-creation at the intersection of lean and agile software development

Master's Thesis
Espoo, 2016

DRAFT! — April 7, 2016 — DRAFT!

Supervisor: Professor Riitta Smeds
Advisor: Otso Hannula M.Sc. (Tech.)

Contents

I	Introduction	4
1	Background and motivation	4
1.1	Rise of agile and lean software development	4
1.2	Capacity for reaction as the differentiator	6
1.3	Why would you have a daily?	7
1.4	Given what we know today, what do we do next?	8
1.5	Dailies and kanban as interrelated practice	10
2	Research problem	11
II	Theoretical framework	12
3	Theoretical approach	12
4	Theoretical research questions	13
III	Empirical study	15
5	Empirical study description	15
5.1	Research context	16
5.2	Setting and participants	16
5.2.1	Environment	16
5.2.2	Roles	17
5.2.3	Tools used	17
6	Empirical research questions	18
7	Data collection and analysis	19
7.1	Research methodology	19

		3
7.2	Overview of collected data	19
8	Empirical findings	20
8.1	The structure of a daily	20
8.2	Use of virtual kanban boards	20
8.3	The team's reactions to new input	20
8.4	Change in the system of work	20
IV	Conclusions	21
9	Results	22
9.1	Understanding the system of work as a shared concep- tual artefact	22
9.2	Renegotiating the system of work	22
9.3	An unfolding conceptualization of the system of work as constitutive of a knowledge co-creation process . . .	22
10	Practical implications	22
11	Evaluation	22
11.1	Limitations of the study	22
Appendix A	Dailies	23
A1	Daily durations	23
A2	Daily segment lengths	24
Appendix B	Kanban boards	25
B3	Lists of kanban boards in chronological order	25
B4	Samples of kanban board structure	25
Bibliography		26

Chapter I

Introduction

1 Background and motivation

1.1 Rise of agile and lean software development

Agile software development practices have become mainstream (West et al., 2010). Among the family of agile methodologies, Scrum introduced by Schwaber (1995) belongs among the most popular (West et al., 2010), being heralded as virtually a de-facto industry standard (Marchenko and Abrahamson, 2008). This research will refer to Scrum as the prototypical example of what it could mean to be agile.

What is it with agile that companies in the IT industry might find attractive? Original motivations for the “new-school” methodologies might have included a desire to figure out simply ways to build better software. Agile can in some respects be taken as antithetical to so-called *waterfall* methodologies which were seen to be a flawed approach. Software development is not like building a house, so attempting to understand and deconstruct the process in a similar

fashion lead to less than stellar results.

Agile process models have been further characterized with simplicity and ease of adaptation as key (Abrahamsson et al., 2002). Agile is represented as the antithesis of dogma, being fit for critical inspection and malleable to use-case specific needs. But if there's no definite way, how do you know you're doing the right things? Adopting agile is indeed not easy when your starting point is a mindset borrowed from the times of the industrial revolution. Agile's very nature seems to have changed with introduction to the mainstream (West et al., 2010), with Marchenko and Abrahamsson (2008) citing problems like "too many meetings" and disciplined effort required to "keep it simple" as present challenges.

To be clear, there is no one single way to run a team or company "the agile way". There is, however, consensus on some things. Necessity of face-to-face interaction is emphasized in agile literature as critical for transfer of ideas and achieving innovative results. (Highsmith and Cockburn, 2002) In the agile manifesto, this was considered important enough for it to take first place on a list of agile values: "individuals and interactions over processes and tools". (Beck et al., 2001) One thing that is undeniably part of the agility toolkit is the practice of a daily: a meeting between development team members taking place every day. "The daily scrum", as described in the leading individual methodology (West et al., 2010), could even be construed to have become a symbol of sorts for agility itself. There is a by-the-book description of what a daily is in the literature, yet owing to the malleability of agile, how the practice is implemented will likely vary significantly. For instance, time constraints of a daily may be relaxed or the general agenda modified if it is found that they do not aid in reaching desired goals (Marchenko and Abrahamsson, 2008).

Another wave of organisations are taking up a lean software development approach, which espouses some of the same practices but presents a different underlying set of principles. Like building a house, neither is software

development like running a factory, yet a brief look into the history of lean will reveal its roots to be in a factory industry. To see where lean fits in the software development scene, it could be thought of as a principled way to figure out better ways to do things.

1.2 Capacity for reaction as the differentiator

One could consider an important difference between agile and lean practice to be the degree of responsivity to new input. As per Scrum, an agile team is run in *iterations* or *sprints* with a fixed agenda. This is based on the idea that *timeboxing* allows a team to focus on essential things needing to be done. A lean process, on the other hand, is built upon the concept of *pull-based flow*. Team members will pull work items for themselves when they are ready to do so, and it is the job of the system built around the team to make sure they have work items to pull in. Fundamentally, a timeboxed team will take into account new input before the start of a sprint, while every instance of a work item being *pulled* into the flow is an opportunity to reflect on new information. A visualization of the flow of work used in lean is called a *kanban board*, a concept related to Scrum's aptly named *Scrum board* which is used to detail a sprint's work items.

A question that both agile and lean attempt to answer in their own ways is "how does one operate in a changing environment". Agile takes a somewhat constrained view by recognizing that when delivering software to a customer, their needs can and will be changed. This can be by virtue of a changing business environment, better understood requirements for the system under development or any number of surprises often so characteristic of projects in real life. The reality of the situation is then accommodated for by building software such that it is amenable to changes in plans with the least amount of effort. Lean's perspective is grander in that it admits for *ways of doing* needing to change as understanding grows and situations change.

Perhaps these two pieces can be taken together to form a discourse. Agile originally offered the promise of ways to build better software, but was easily oversold as a silver bullet, leading to failures in adoption and disillusionment. Clearly there was something positive there, but how could *our* organisation make use of it? The path to agile adoption was anything from clear. Enter lean software development: with the endorsement of ideas like “start where you are” and “continuous improvement”, one could conceptually figure out the *path* to enlightenment instead of the ostensible *destination*.

1.3 Why would you have a daily?

In section 1.1 we made the assumption of a team having a daily due to it being part of the package ascribed by the Scrum playbook. Indeed, cargo cults where the participants are not aware of why they choose their methods exist in the industry. However, taking a note from the previous section and admitting that ways of doing can be questioned, let’s assume that the team is equipped with a sense of reflexivity sufficient to ask the very reasonable question: why would you have a daily?

It is easy to come up with arguments on how the daily might not be an effective use of its participants’ time. They are synchronous, meaning participants must be present simultaneously. Though the recommended time box for a daily is 15 minutes, this length limit does not go without saying (as we will observe in section 8.1). Throughout that time, no code is written, no bugs are fixed, no features are implemented, and no customer is served. There are very real opportunity costs associated with having a daily. Making the assumption that the daily is not mandated by external factors (such as perhaps an external consultant insisting on a strict adherence to Scrum), there must be some other value that the team sees in spending that time.

What is the value seen by the team? It must be that the team itself has

a hypothesis, an understanding of the value sought, or it would not choose to have dailies. For a team aware of the costs, making the contrary choice would be deliberately wasteful, so let us further assume that not to be the case. The people engage in something that has no value in terms of software development work done, but is ostensibly worthy of spending everyone's time on *every day*. Because no tangible "work" gets done, it follows that the time left for that work afterwards can hypothetically be spent more effectively. The daily's role must be in somehow enhancing the organisation's effectiveness, meaning capability for value-adding service delivery.

1.4 Given what we know today, what do we do next?

Continuing from the line of thought started in 1.2, we will consider another argument having to do with the handling of new input. This is done in the hopes that we will be able to see the daily and the use of kanban as related through serving the same needs.

Software development is not done against a static, unchanging background. This is indeed the basis on which modern methodologies build upon and where they derive their usefulness from, as discussed in 1.1. We can say, at the very minimum, that situations change in the world outside somehow, and that to optimize its behavior for yielding better outcomes an organization would do well to take into account these changes. But here the aspiring kanban practitioner may be faced with a dilemma. The world changes a lot; there is at any given point a potentially unbounded queue of input to process, against a very decidedly bounded capacity for handling that input. A perfect system will absorb any and all input to make perfectly informed decisions on what to act upon. Alas, the world is not perfect, and we will not be stopping every day to reconsider everything possible in the light of new input. Let us call this the bounded rationality dilemma.

As discussed in 1.2, a kanban system is on one respect a visualization of work and as such serves to answer the question of *how things get done*. Taking into account the bounded rationality dilemma, the system needs to be accompanied with an answer to the related but subtly different question of *what is a thing needing to get done*. Consider the crude example of a web based service's production servers being literally on fire. This is surely a situation where a team will be reaching for the fire extinguisher without the question of consulting a kanban board. Indeed, situations do arise where taking prompt action is crucial; in other situations it is not at all clear what the salient points in the input are and what there is to be done. There are literal fires, obvious to any human being, but what is to be made of figurative fires and what would give one the ability to tell them apart from entirely benign or even meaningless events?

Let us briefly go back to lean and apply the same question to kanban as we did for dailies. Why visualize work, and why use kanban especially? Of course, kanban being a thing that takes effort and time to execute yet not being software development per se, the line of argumentation we just followed could be found compelling here as well. With no further considerations needed we can conclude that they must both be supportive of software development. Given that conclusion, what is the relation between the two?

Consider a daily with an agenda defined by the Scrum sprint's goals. We would think that when the agenda is fixed and can be known to parties beforehand, the discussion and actions undertaken will be directed towards getting those things done in the fashion the team believes to be best. "I did X yesterday, I'm going to do Y today and will be needing help with Z" would be an apt description of a common schema that each team member could use their chance of communicating at the daily for. This is, indeed, part of recommended best practice. Yet with us now having a way to tell exactly what the mentioned X, Y and Z are by taking a look at the visualization of our work, is not the daily rendered redundant?

What if we consider the situation in a lean workflow when there might be any amount of new input for the team to absorb? Fresh customer insight, feature requests, bug reports and detected anomalies in running production systems are a few examples of what a team might want to decide on reacting to. The space of things to discuss and potential actions to take is vastly larger, giving rise to the possibility for a very different and arguably more interesting daily to emerge. The researcher would argue that instead of kanban making the daily redundant, the combination of both is instead worthy of investigation.

1.5 Dailies and kanban as interrelated practice

The previous section argues that the intersection of a daily and kanban might have interesting features. To rephrase section 1.3, the researcher finds that the only reasonable hypothesis would be that the development of the software is somehow made better through this: that there's some means, some knowledge that is gained through the engaging in a daily that at least plausibly more than makes up for the time spent.

What is missing a description of *how* the daily is of value. To avoid a circular argument or a *begging the question* fallacy, we should avoid reaching for this description in the very literature that recommends it. So, how do we describe the daily? We would like to come up with a way to describe the daily as a recurring event where knowledge is produced, made better or different in some way. In doing this we would find ourselves capable of not only showing how a real-life daily *is*, but deriving a description for the *use* of such a daily from first principles. On the corollary, failing to do so would shed some doubt on the fundamental usefulness of the activity.

Drawing from section 1.2, if being reactive to new input is important to software development efforts, the daily may be seen as a *sensory organ* for

the software development organisation to feel its way forward in a chaotic and ever-changing landscape. Both agile and lean discourses have their own ways of discussing and arguing about these things, of course. The researcher would prefer to have at hand a more universal, a less context-dependant tool of evaluating the kind of organizational-learning-through-knowledge-work at hand here. Luckily, there exists a branch of applied sociology called *organizational research* and within that branch resides something called the *practice-based approach* (Nicolini et al., 2003).

The research conducted in this paper aims to describe the practice of a daily in the context of lean software development, give a rich account of it through a case study and connect the practice to larger themes of knowledge co-creation and organisational learning. We want to elucidate both how a daily happens in practice, and what the practical outcomes are, or in other words: against the background of lean principles, *what exactly is the use of a daily?*

2 Research problem

RP: How can the combination of daily and kanban practices support software development?

Chapter II

Theoretical framework

3 Theoretical approach

In my thesis I will review literature from software development process, organizational, and sociological research to create a theoretical synthesis.

Depictions of agile methodology adoption, practice and evolution:

- Rise of agile into the mainstream and the abandonment of agile orthodoxy (West et al., 2010)
- Agile practice and culture (Sharp and Robinson, 2004; Robinson and Sharp, 2005; Robinson et al., 2007)
- Challenges in agile adoption (Nerur et al., 2005; Marchenko and Abrahamsson, 2008)

(Problem: Most “agile” literature seems to take the perspective of Scrum, XP et al, where I’d be more interested in a Kanban & Lean perspective.)

Understanding the agile organization as an evolving sociotechnical system:

- Roots of agile principles (Vidgen and Wang, 2006; Nerur et al., 2010)
- Systems of problems and interactive planning (Ackoff, 1997)

Characterization of scrum / kanban boards or virtual ones:

- The role of story cards (Sharp et al., 2006)
- Visualizations as objects of continuously unfolding epistemology (Ewenstein and Whyte, 2009)
- Boundaries as enablers of and barriers to innovation (Carlile, 2002)

How might new information be created by manipulating aforementioned artefacts:

- New knowledge creation by intersubjectively accepted novel distinctions (Tsoukas, 2009)

4 Theoretical research questions

Our search for theoretical understanding required for answering the research question takes a three-pronged approach.

TRQ1: What is a daily?

TRQ2: What is a kanban system?

TRQ3: How can the uses of daily and kanban be seen as practices?

To successfully make claims about dailies and kanban, an understanding on both is required. Questions *TRQ1* and *TRQ2* are aimed at setting solid ground based on which we can find the activities observed in the empirical study as being representative of the *use of a daily* or the *use of a kanban system* respectively. Following up on that, with *TRQ3* we set out to find a

way to describe dailies and kanban through a theoretical lens that can serve as a synthesis and provide vocabulary for describing both and either of them. Through this unified lens we will gain the ability to inspect them and their relation as being overlapping or simultaneous in nature, as will be required in the empirical analysis section.

Chapter III

Empirical study

5 Empirical study description

The researcher is interested in gaining a deep understanding of the practice of a daily as it occurs in the context of software development. The agile discourse has a history with ethnographically-inspired studies of agile practices (Robinson et al., 2007; Marchenko and Abrahamsson, 2008), and a similar empirical approach was chosen here. An ethnographical approach is specifically well suited for discovery of how practitioners behave within and use the practice under investigation. Given the choice of theoretical background where a sociological understanding of practice plays a significant role, and the desire to observe in context, not to interfere with and to improve, ethnography makes for a sound choice. In the sense that the research also considers parts of speech and specific manipulations of cognitive artefacts as constitutive of knowledge co-creation, the empirical study also carries microethnographical tones (Streeck and Mehus, 2005).

5.1 Research context

The research was conducted as a case study of a Helsinki-based software startup in the mobile B2B industry with a headcount of approximately 20 people and a development team consisting of at most 10 at the time of the research. The development team's habit of gathering around a collection of virtual kanban boards every morning was taken as an opportunity to learn about the practice of dailies as exercised within the context of a lean workflow. The researcher took part in the development team's dailies wearing two hats: the ordinary team member hat, which necessitated normal interaction with the team within the context of the daily, and a researcher hat which involved making observations and logging the proceedings.

5.2 Setting and participants

5.2.1 Environment

The chronological context of the development team daily was after a short whole-company standup meeting scheduled to start at 10 in the morning every day. The discussion from the standup meeting would sometimes anecdotally have an impact on the content of the development team daily, but expanding on this relation was not part of the research objectives.

The daily meetings were invariably held in a meeting room equipped with a table, a couch, a TV screen to which laptop computers could be plugged to, and two whiteboards. Attendees would customarily be seated for the duration of the daily. There was not enough room for more than 8 people to sit, so in the case of meetings which more than that amount attended, some would be forced to stand or choose to sit on the floor. On the table, two people would have their laptops open: the *driver* and the *secretary*.

5.2.2 Roles

The driver connects their laptop to the TV screen, enabling the team to view the virtual kanban boards and items therein. Throughout the daily the driver will switch between kanban boards and task items according to how the team's discussion proceeds.

The secretary affects decisions made by the team. This can take the form of manipulating tasks on the boards, making modifications to board structure, and making meeting notes.

The driver and the secretary are the only explicit roles defined in the context of the daily. Other than that, the participants consist of developers, quality assurance, technical leads and product owners. All of the development team, including quality assurance but not necessary technical leads, were compelled to take part by convention. Product owners were present sporadically.

5.2.3 Tools used

The most important tool made use of was Asana, a web-based task management application. It views work as *projects*, which consist of *tasks* in a linear list, which can be delineated into segments with *labels*. Tasks consist of a title and a description, may be assigned to individuals, may be given tags, and may be commented on. The daily relied on Asana as a virtual kanban board tool. The projects are understood as *boards*, and the labels are understood as *states* that a task can have.

In addition, meeting minutes were published through Flowdock, a company-internal instant messaging and group chat application. The application structures communication in terms of *flows*, under which participants may converse textually. One of these flows was dedicated for meeting minutes, so

that there was usually no other discussion besides what was logged by the secretary.

Tools for sharing knowledge in the physical environment, such as post-its or the whiteboard present in the room, were seldom touched.

6 Empirical research questions

The following research questions are posed to the empirical research data.

ERQ1: How is the organisation's system of work as understood by the individual team members manifested in the daily practice?

ERQ2: How does the team's understanding of the system of work change?

ERQ3: How does the daily practice support co-creation of knowledge?

The first question is aimed at discovering whether the daily practice in fact constitutes the use and manipulation of the system of work as a shared conceptual artefact. The second question is needed for making out if the team actually improves upon the conceptual artifact, linking it to the third question analyzing the role of the daily practice as an organisational mechanism for knowledge co-creation.

7 Data collection and analysis

7.1 Research methodology

The data for this research was collected through the use of audio recordings. The events observed were part of the everyday proceedings of the software development team. The team was informed beforehand of the intent to record the proceedings and that the intent would not be to interfere with what occurs naturally or deploy interventions for purposes of study. The author took part in all of these events and took the same role of secretary (as described in 5.2.2) in each one. The dailies took place in a meeting room that was fitted with a table, on which a mobile device was set for purposes of recording before the beginning of each daily. Notes on what went on in each daily were made to accompany the recordings as basis for further analysis later. In addition to audio recordings, the kanban boards' structure was captured as screenshots from the virtual kanban board tool described in section 5.2.3.

7.2 Overview of collected data

From December 2015 to January 2016, a period of time including a break for seasonal holidays, 20 individual daily events were observed and recorded. The duration was chosen both to get a representative sample of dailies and to allow for the potential discovery of a shift in the team's understanding of the system of work and the flow of work through the system. It was initially hypothesised that this change would most concretely be exemplified in the virtual kanban boards' structure, and for reflecting this hypothesis the visual structure of the boards was captured for later examination.

The 20 dailies recorded altogether contain a total of 11 hours, 59 minutes and 12 seconds of audio. Average duration for a daily was 35 minutes and

58 seconds, with a minimum at 11 minutes 57 seconds and a maximum at 1 hour 21 seconds. Readily it can be observed that these dailies were not conformant to the Scrum-espoused ideal of a maximum of *15 minutes* per daily. Complete tables of duration data can be found in appendix section A1.

8 Empirical findings

8.1 The structure of a daily

8.2 Use of virtual kanban boards

8.3 The team's reactions to new input

8.4 Change in the system of work

Chapter IV

Conclusions

9 Results

9.1 Understanding the system of work as a shared conceptual artefact

9.2 Renegotiating the system of work

9.3 An unfolding conceptualization of the system of work as constitutive of a knowledge co-creation process

10 Practical implications

11 Evaluation

11.1 Limitations of the study

Appendix A

Dailies

A1 Daily durations

Daily number	Date	Duration
01	8.12.2015	36:36
02	9.12.2015	48:53
03	10.12.2015	28:30
04	11.12.2015	33:51
05	14.12.2015	28:16
06	15.12.2015	17:47
07	16.12.2015	27:08
08	17.12.2015	20:26
09	18.12.2015	19:03
10	21.12.2015	31:13
11	22.12.2015	11:57

12	7.1.2016	34:28
13	8.1.2016	47:33
14	11.1.2016	36:06
15	12.1.2016	51:44
16	13.1.2016	1:00:21
17	14.1.2016	53:35
18	15.1.2016	40:29
19	18.1.2016	49:57
20	19.1.2016	41:11

Table A.1: Durations of daily recordings by daily number and date.

Metric	Duration
Total	11:59:12
Average	35:58

Table A.2: Aggregate daily duration data.

A2 Daily segment lengths

Appendix B

Kanban boards

B3 Lists of kanban boards in chronological order

B4 Samples of kanban board structure

Bibliography

- P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta. *Agile software development methods. Review and analysis*. VTT Technical Research Centre of Finland, 2002.
- Russell L Ackoff. Systems, messes and interactive planning. *The Societal Engagement of Social Science*, 3(1997):417–438, 1997.
- K. Beck, M. Beedle, A. Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, B. Kern, J Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, and D. Thomas. Manifesto for agile software development. 2001. URL <http://agilemanifesto.org>. Noudettu 2010.10.23.
- Paul R Carlile. A pragmatic view of knowledge and boundaries: Boundary objects in new product development. *Organization science*, 13(4):442–455, 2002.
- Boris Ewenstein and Jennifer Whyte. Knowledge practices in design: the role of visual representations as 'epistemic objects'. *Organization Studies*, 30(1):07–30, 2009.
- J. Highsmith and A. Cockburn. Agile software development: The business of innovation. *IEEE Computer*, 34:9(9):s. 120–127, 2002. ISSN 0018-9162.
- A. Marchenko and P. Abrahamsson. Scrum in a multiproject environment: An ethnographically-inspired case study on the adoption challenges. In

- Agile, 2008. AGILE '08. Conference*, pages 15–26, Aug 2008. doi: 10.1109/Agile.2008.77.
- Sridhar Nerur, RadhaKanta Mahapatra, and George Mangalaraj. Challenges of migrating to agile methodologies. *Communications of the ACM*, 48(5): 72–78, 2005.
- Sridhar Nerur, Alan Cannon, VenuGopal Balijepally, and Philip Bond. Towards an understanding of the conceptual underpinnings of agile development methodologies. In *Agile Software Development*, pages 15–29. Springer, 2010.
- Davide Nicolini, Silvia Gherardi, and Dvora Yanow. *Knowing in organizations: A practice-based approach*. ME Sharpe, 2003.
- Hugh Robinson and Helen Sharp. Organisational culture and xp: three case studies. In *Agile Conference, 2005*, pages 49–58. IEEE, 2005.
- Hugh Robinson, Judith Segal, and Helen Sharp. Ethnographically-informed empirical studies of software practice. *Information and Software Technology*, 49(6):540 – 551, 2007. ISSN 0950-5849. doi: <http://dx.doi.org/10.1016/j.infsof.2007.02.007>. URL <http://www.sciencedirect.com/science/article/pii/S0950584907000110>. Qualitative Software Engineering Research.
- K. Schwaber. Scrum development process. In *OOPSLA Business Object Design and Implementation Workshop*, pages 10–19, 1995.
- Helen Sharp and Hugh Robinson. An ethnographic study of xp practice. *Empirical Software Engineering*, 9(4):353–375, 2004.
- Helen Sharp, Hugh Robinson, Judith Segal, and Dominic Furniss. The role of story cards and the wall in xp teams: a distributed cognition perspective. In *Agile Conference, 2006*, pages 11–pp. IEEE, 2006.
- Jürgen Streeck and Siri Mehus. Microethnography: The study of practices. *Handbook of language and social interaction*, pages 381–404, 2005.

Haridimos Tsoukas. A dialogical approach to the creation of new knowledge in organizations. *Organization Science*, 20(6):941–957, 2009.

Richard Vidgen and Xiaofeng Wang. Organizing for agility: A complex adaptive systems perspective on agile software development process. 2006.

Dave West, Tom Grant, M Gerush, and D D’silva. Agile development: Mainstream adoption has changed agility. *Forrester Research*, 2:41, 2010.