Aalto University
School of Science
Degree Programme in Information Networks

Eevert Saukkokoski

What exactly is the use of dailies:

The daily as a practice in the context of lean software development

Master's Thesis
Espoo, 2016

DRAFT! — April 20, 2016 — DRAFT!

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

Contents

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 Abrahamsson, 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 though 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 as practice

The previous section argues that the use of a daily in conjunction with kanban might have interesting features. To rephrase section 1.3, the researcher finds that a 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. 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. In other words, it helps to answer the question of *how do we know to do the right things*.

Disregarding for now the question of *whether* there's evidence for this, what is missing is a method of description for *how* the daily works to do what would be purported here. Both agile and lean discourses have their own ways of discussing and arguing about their respective benefits, of course. To avoid a circular argument or a fallacy of *begging the question*, 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.

The question of knowing how to do the right things brings us to the question of knowledge, as well as that of doing. What is knowing, and what is doing the right thing? How is the concept of the right thing to do formed and how is it put to action? In a society increasingly relying on expert knowledge, it comes as no surprise that there exist several fields of research impinging on these subjects. We will be looking at knowing and learning through the theoretical lens of the sociological concept of *practice*.

The *practice-based approach* of organizational research looks at knowledge as acquired in participation, or knowing as something that is inseparable from doing (Nicolini et al., 2003). Practicing can be viewed as a transformative process: not only is there an equivalence between knowing and practicing, but knowledge transforms itself through use (Gherardi, 2011). Practices are social institutions that are sustained by their use within a sociomaterial context. They exist in a continuous state of becoming and responding to an unfolding reality, whereby learning can be understood as a recursive process of interaction between the practice and its practitioners.

The research described in this paper aims to describe the daily as a practice in the context of lean software development. Account of the practice will be given through a case study, elucidating both how a daily happens and what the outcomes are. In other words: against the background of lean principles and through the lens of practice, *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 in agile software development?

TRQ2: What is a kanban system in lean software development?

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

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 and the secretary are the only explicit roles defined in the context of the daily.

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.

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 does the daily practice reflect the activity system as understood by the team?
>
> **ERQ2**: How does the exercise of a daily improve upon the daily practice itself?
>
> **ERQ3**: How does the daily practice interact with other software development practices?

The first question concerns the team's understanding of the activity system that the daily practice is embedded in. We would like to observe how the activity system appears within the daily. How does the sociomaterial context of the daily support the conceptualization of the activity system?

The second question considers the "practice-nature" of the daily. When the team engages in a daily, does it evolve? How is the logic of action redefined – can we see emerging disturbances to the activity system being settled within the daily?

With the third question we probe the boundaries of the daily practice and attempt to discover its interfaces with other practices identifiable in the exercise of software development. Can we see if the daily somehow informs other practices or aids in their execution?

The undertaking is, altogether, to show that the activity system plays a role in the daily, that the daily serves to improve on the team's understanding

of the activity system and that this understanding is linked to the exercise of creating software. Should we succeed, we will be able to point to the use of a daily residing within this continually developing understanding of the activity system, or put more simply, *improving on how the work works*.

7 Data collection and analysis

7.1 Research methodology

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 time span 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.

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.

Audio recordings consitute the main body of research data. After dailies, notes on what went on were made to accompany the recordings as basis for further analysis later. In addition to audio recordings and the accompanying notes, the kanban boards' structure was captured as screenshots from the virtual kanban board tool described in section 5.2.3.

Audio recordings were transferred from the recording device, an *iPhone 5S*, as *mp4* files, and played back using *VLC Player*. An initial analysis and structuring of the material was carried out for each daily by first reviewing the associated notes as a clue on possible interesting events, features, or background information useful for understanding the proceedings and then playing back the audio in a linear fashion. Notes of the audio were made in text format, logging timestamps of discussion topics and interactions that were found to be of interest. This *segment analysis* yielded an outline of dailies that is further described in section 7.2.2.

Capturing the kanban boards' structure as screenshots yielded material of two different kinds. The web software Asana described in 5.2.3 presents what the researcher understood as kanban boards as a linear list. The contents of this list were extracted and are presented as indexes in appendix section B2. The individual kanban boards' structure, expressed likewise as a linear list of labels or 'states', was extracted in a similar fashion and presented fully in appendix section B3.

7.2 Overview of collected data

This section presents an outline on the data and its salient features. Table III.1 is introduced as a basis for reference. When specific dailies are discussed, their index number (from `01` to `20`) will be used.

Daily index	Date	Recorded	Effective	Language	Attendance
01	8.12.2015	36:36	34:01	English	7
02	9.12.2015	48:53	47:19	English	?
03	10.12.2015	28:30	27:38	English	7
04	11.12.2015	33:51	32:25	English	9
05	14.12.2015	28:16	27:34	Finnish	7

06	15.12.2015	17:47	15:23	Finnish	5
07	16.12.2015	27:08	26:42	Finnish	5
08	17.12.2015	20:26	19:50	Finnish	?
09	18.12.2015	19:03	17:20	Finnish	?
10	21.12.2015	31:13	30:10	Finnish	4
11	22.12.2015	11:57	11:07	Finnish	3
12	7.1.2016	34:28	32:14	English	8
13	8.1.2016	47:33	45:52	English	7
14	11.1.2016	36:06	35:16	English	10
15	12.1.2016	51:44	51:03	English	9
16	13.1.2016	1:00:21	59:36	English	8
17	14.1.2016	53:35	52:02	English	?
18	15.1.2016	40:29	40:00	English	7
19	18.1.2016	49:57	48:25	English	8
20	19.1.2016	41:11	39:24	English	8

Table III.1: Overview of audio material on dailies by daily index number and date. *Recorded* is the duration of the audio recording. *Effective* is the duration of the part of the recording constituting the daily activities (see A1). *Attendance* is the number of participants present.

7.2.1 Attendance and language

The dailies gathered an attendance of three to ten participants. As table III.1 shows, there is a drop in attendance halfway through the observation period.

This is attributable to national holidays and the associated vacation periods starting in a staggered fashion. The team's mean daily attendance during normal operation appears to be seven to eight participants (dailies `01` to `05` and `12` to `20`).

Attendees' overall roles in the team could be divided into four categories: *developers* working on the product directly, *quality assurance* responsible for testing the product and verifying that requirements are met, *product owners* of managerial import capable of making backlog priorizations, and *management* whose opinion carries great weight in the way of conducting operations (such as the daily itself). Developers and quality assurance were by convention compelled to take part (at the absence of a good reason to the contrary), whereas product owners and management were not. Roles, e.g. that of product owner and management, may overlap; in this case the more situationally relevant label will be used per transcribed interaction.

For the sakes of both succinctness and preservation of anonymity, when these roles are referred to in transcriptions the following labels will be used. `D` for developers, `QA` for quality assurance, `PO` for product owners, and `M` for management. Different people will be differentiated by suffixing the label with an index number, such that e.g. `D3` will refer to the third developer taking a turn of speech through the specific interaction. Indexes used are specific to the interaction, as opposed to assigned per individual.

Due its international composition, the team handled most of the dailies in English, reverting to Finnish when team member attendance allowed for it. Seven out of the twenty dailies observed were conducted in Finnish (dailies `06` to `11`). When quotes originally in Finnish are presented in the empirical findings section, translations by the researcher are shown for accessibility.

7.2.2 Durations and segments

The 20 dailies recorded altogether consist of a total of 11 hours, 59 minutes and 12 seconds of audio. According to the segment analysis the recordings could be trimmed to a total effective duration of 11 hours, 33 minutes and 21 seconds constituting the entirety of what was considered as the daily activities. The average effective duration of a daily was 34 minutes and 40 seconds.

Metric	Effective
Total	11:33:21
Average	00:34:40

Table III.2: Aggregate metrics of effective daily durations.

A total of 13 different daily segments were identified in the material. Table III.3 shows a summary of the data. Complete tables of the daily segments observed can be found in appendix section A1.

Table III.3: Total durations of identified segments, their average length and counts of their occurrences in the material.

Segment	Sum	Average	Count
INBOX	04:04:35	00:11:39	21
FIRES	02:18:41	00:06:36	21
INTRO	01:46:12	00:05:54	18
OUTBOX	01:21:28	00:04:04	20
PRIORITY LANE	00:24:17	00:01:26	17
QUESTIONS	00:23:40	00:01:19	18
NON-FUNCTIONAL CONTINUUM	00:23:25	00:01:57	12
BOARD OVERVIEW	00:20:38	00:06:53	3

Segment	Sum	Average	Count
CYCLES	00:15:20	00:07:40	2
NON-CORE MODULE CONTINUUM	00:13:25	00:01:41	8
MONEYBOX	00:00:49	00:00:25	2
MODULES CONTINUUM	00:00:25	00:00:25	1
MARKETING BOX	00:00:13	00:00:13	1
WATBOX	00:00:13	00:00:13	1
Total Result	11:33:21	00:04:47	145

The segments identified bear some elaboration. Unless otherwise specified, the segment is named by reference to a concrete instance of a board. Prominent examples include `INBOX` and `FIRES`. The exceptions are enumerated below.

Most dailies start with an `INTRO`. The `INTRO` is a segment where the team has engaged the daily but is not yet focused on any of the boards available. The segment can be prompted by a call such as *"So, general things"* (`01`, 02:02), or it might begin more fluidly e.g. with a team member presenting a topic for discussion as in this exchange from daily `03` (00:40):

- *"That's an awesome pear."* (unstructured discussion, indistinct background chatter)
- *"Should we have a new nickname for [team member]?"* (a topic is presented)

Some dailies include a `BOARD OVERVIEW`. In these segments, the focus is not on individual boards, but their overall status or health, relationships and priorities between them and the arrangement of them in the list of boards. Here's an example where a developer inquires after the purpose of colored labels used for different boards (daily `03`, 26:21):

- *"What about.. are the, uh, colors?"* (developer asks question)

- *"Yellow is in QA. Purple has passed QA. Then there's grey that is in smoketest and, well, you won't probably ever see the blue because I.. blue means it's.. has been smoketested and I'll archive it any moment now."* (QA responds)

Most boards used in dailies were ones of persistent nature. They are not explicitly bound to dates or weeks or deployment cadences. `CYCLES` make an exception to this. They are boards that are numbered and constitute a bundle of shippable things. Boards labelled with the same number would be deployed together, and there are a multitude of them in use at the same time. As alluded to in the quote by *QA* above, the board for a specific cycle will vanish from sight by being archived after work in it has been completed. All of these were grouped together as `CYCLES`. Discussion generally involves *what is being worked on*, *by whom* and *when it will be completed*. An extract from daily `02` (12:58) demonstrates how conceptions on progress of work and implications of handovers are negotiated in context of `CYCLES`:

- *"So, are you still going to make it by friday?"* (QA prompts)
- *"Yes"* (response by developer)
- *"Still confident"* (QA confirms)
- *"Quite... I have to, I'm leaving on friday"* (another developer interjects)
- *"So who will be fixing your stuff on monday when it gets to QA or tuesday when it gets to QA?"* (QA expresses concern)

Cycle is the team's name for a single development cadence. It is used for sets of features that are completed in synchrony with each other and that cannot be shipped incrementally. The opposite of this is a *continuum*, evident in boards such as `NON-CORE MODULE CONTINUUM` and `NON-FUNCTIONAL CONTINUUM`. Work done here can generally be shipped to production immediately after completion.

8 Empirical findings

This section presents empirical findings based on which we will answer the empirical research questions from section 6. To settle **ERQ1**, in 8.1 we describe the structure of a daily and link this structure to the use and arrangement of kanban boards.

8.1 How is the daily structured?

This section describes the structure of a daily as it appears based on the segment analysis described in 7.1. No two dailies followed exactly the same structure, but there were clear tendencies to be observed. Let us consider daily `03` as a prototypical example, as it has close to average duration, attendees and a clear segment outline. The outline can be seen in table III.4.

Segment	**Duration**
INTRO	00:10:45
INBOX	00:01:45
FIRES	00:04:05
PRIORITY LANE	00:03:30
OUTBOX	00:03:20
QUESTIONS	00:02:00
BOARD OVERVIEW	00:02:13
Total duration	00:28:18

Table III.4: Outline of daily `03`, presented here as a typical daily. Excerpt from appendix section A1.

8.1.1 Typical segments in depth

8.1.1.1 Introduction The daily starts with an `INTRO` segment after making sure everyone is accounted for. It's noted that a team member has been absent for a while and that is affecting the team's capability of responding to fires. The team also takes issue with the state of communication between them and another team.

02:30 "Apparently it seems to be some ... hard to reach between business and marketing now, so–" (`D1`, referring to events from before the daily)
"You mean sales and marketing" (`M1`, correcting `D1`'s expression of the overall team structure)

02:41 "I'm just saying that we're not alone" (`D1` referring to difficulties shared with the team on a previous occasion)

02:58 "And I felt, it was like .. difficult" (`D1` starting to relate the experience)

03:01 "There's like .. two walls in between [..]" (`D2` continuing the sentiment metaphorically)

03:10 "So `M1` can you take down the walls or?" (`D3` asking for help in resolving the difficulties)
"Yes yes. `M2` comes back, we will shuffle the rooms anyway" (`M1`, referring to concrete workspace arrangements)

We can observe that the exchange is rich in history and context. Discussions from what appear to be previous dailies are referenced, as well as more immediate occurrences from the same day. The team seems content to describe the situation on a very abstract level, which could point to an understanding of concrete experiences already having been shared. The tension appears to be relieved by an explicit plea for help and a corresponding assurance that very concrete actions will be taken, despite the metaphorical level of discussion so far. Taken together, we see that the team uses the opportunity for coming together to discuss their woes and to find ways for recourse.

The team proceeds through an unstructured go-through of worries until arriving at what is apparently of immediate relevance: an upcoming feature will be requiring a significant amount of quality assurance after being shipped to production. The team assesses ways of approaching this, with `QA` and `M` negotiating the scope of testing required and the dependencies between work items that will have to be resolved to get to this stage.

08:28 "Done today then?" (`M`)

"We can't do that, the thing is not in production [..]" (`QA`)

08:35 "So what is blocking that?" (`M`)

"Uhh .. the [feature] hasn't passed QA" (`QA`)

"Okay. And what is blocking that?" (`M`)

08:44 "`D` has fires." (`QA`)

"No, they're working right now, there's no [feature] related tasks" (`D`, challenging `QA`)

"Yes, there is." (`QA` restating their position)

This exchange appears to relate to the status of a specific cycle, but the team is not using the cycle's board to mediate the discussion. Thus the exchange is part of the `INTRO` segment, not `CYCLES`.

The team continues to negotiate the interrelations between work items handled by different team members and handovers required. Concern is expressed about a team member leaving for vacation after the ongoing week and some items perhaps not being completed by then, but reassurences to the contrary appear to placate everyone.

An attempt is made to start the `INBOX` segment with the prompt:

10:05 "What about starting from Asana now?" (`M`)

This is ineffective, because another team member wants to discuss handovers not yet mentioned. The secretary can be heard making meeting minutes on

the subject.

8.1.1.2 Inbox The `INBOX` segment finally starts at **11:25**. The *driver* opens Asana and allows the team to view items in the inbox on a TV screen. The inbox is a board that contains new items as input to the team and to be handled in the daily proceedings. Starting from the topmost item, item labels are read out loud. Most often, a variation of the following question is used to settle the imminence of items encountered:

11:51 "Fire or not?" (`M`)

Either the item is a fire, requiring the team's immediate attention, or it can be postponed to a later time. Fire would mean that the *secretary* moves the item for it to get handled in a later `FIRES` segment. The team members will respond with either "fire" or another board where the item would get handled in the appropriate fashion.

12:23 "Not a fire?" (`M`)
"No .." (`D`)
"Wishlist." (`M`)

The wishlist is a labeled section of the inbox board that is intended for a later grooming with relevant product owners involved. The secretary thus moves the item away from the inbox's topmost segment to the wishlist, and continuing in this way the inbox is gradually consumed of unhandled items.

8.1.1.3 Fires Once the inbox is empty, the driver without prompt moves the team's view to a board titled *"[FIRE]: This must be emptied"*, signaling the start of the `FIRES` segment. Again the team will proceed from top to bottom, this time discussing items that have moved between states since last observation.

Based on the board's labeling of states, (detailed in appendix B3), the items further to the top are closer to having been resolved whereas at the bottom no work is being done on them. The more than a dozen different states communicate an intricate understanding of a work item's progression from backlog to development, to testing, production and finally a state of doneness that can be agreed on by the team.

Distinguishing this segment from the previous one, items will usually not be moved by the secretary. They may, however, be marked as complete after the team agrees there's nothing more to be done.

Progress with fire extinguishing, or lack thereof, can be touched briefly:

14:32 "So `D1`, all ok, need anything?" (`QA`, referring to work item presented by driver)
"Uh ... yeah, no, just to note that-that me and `D2`, we had a, had a prepared battle plan for this specific occasion, and our first attempt at-at fixing it instantly failed. So .. I'm going to have to *actually* fix this." (`D1`)

14:52 "Can you actually fix this?" (`M` seeks confirmation)
"Yes, I can actually fix this." (`D1` concludes)

In this case the details of how the actually-fixing would be done are omitted in the discussion as something that belongs outside the scope of the daily. What gets communicated is the belief that work items will be moving today despite lack of progress thus far.

Sometimes a balance between expedited technical implementation and other rationale, such as product design, needs to be reconsidered. Here, a developer presents a caveat in their intended approach to resolve a fire:

15:08 "Most likely the fix is kind of .. people don't like it, but like that's the .. easiest way to fix" (`D`)

"What?" (`QA`, incredulously)

"You need to save the changes before you can open the [feature]" (`D`)

15:22 "Why doesn't it save it automatically? [..]?" (`M`)

"It did, but–" (`QA`)

"No, it didn't" (`D`)

"It looked like-!" (`QA`)

"Yeeah it looked like, yeah, sure. (short laughter)" (`D`)

15:33 "But why not that way?" (`M`)

"Well, we could .. do that that way" (`D`)

15:37 "Wouldn't that make more sense?" (`M`)

"Yeah." (`D`)

"Just saves it. Next." (`M`)

Negotiation on how thoroughly the board will be combed also happens. Here, a developer poses the question of whether *backlog items* (meaning items that have not yet moved since their addition to fires) should be handled at this time:

16:42 "Do we want to discuss the backlog items?" (`D1`)

"They are the same, I guess ... Not moved" (`D2`)

"But no new fires. Or are these two by `D3`, are these new?" (`D1`)

"No no, they were [new] yesterday" (`QA`)

It's shortly determined that there would be nothing to gain, and the team moves on to a new board.

8.1.1.4 Priority lane The driver opens up *"Priority lane"*, commencing the `PRIORITY LANE` segment. The flow of discussion is similar, with moved work items getting discussed from top to bottom. However, the team has chosen to label the states differently. There is significantly less structure as compared to fires: **DONE**, **DOING**, **TODO** and **LONGSTANDING**. This can be attributed to the non-technical or perhaps ad-hoc nature of work

items placed here.

A developer immediately starts relating the outcome of a discussion that was modelled as a work item on the priority lane and can now be seen by the team as "done":

17:15 "This was huge: [..]" (D, with explanation of defects discovered in production system)

Due to the critical nature of the findings, the team would expect there to have been a follow-up work item. A manager expresses concern on whether action will be taken. This gives rise to a discussion on how the flow of work is being modelled for the team's benefit in this specific occasion:

18:00 "But isn't that kind of a fire?" (M)

18:08 "I'm doing everything I can, I have the one other fire first." (D)

"Yeah but where is this kind of task?" (M)

"It's in the module continuum, it's being worked on .. like, right now .. there's .. this many tasks about it" (D)

"Okay ... but wouldn't it make sense to promote them as a fires also? And not leave them into the module continuum?" (M)

"Hhh if you want to-" (D)

"For more visibility ... because you can have task in two projects" (M)

A rationale of visibility is presented, along with a bargain in that the developer's existing modelling can remain essentially undisturbed. It also becomes evident how the team has a conception of fires as *urgent-but-unplanned* versus *planned-and-prioritized*, which results in a curious tension:

18:48 "Yeh ... which one do you want [to see in fires]?" (D)

"Everything that's considered a fire [and is] in module continuum ... because that's essentially blocking the module continuum for continuing ... on the actual *planned* features" (M)

19:04 "These are the planned features .. and fixes. These are what, uh, what we have prioritized with `PO`" (`D`)

The tension is resolved by refining the modeling to include the "fire-nature" of the work items and showing them in fires. The tension's root cause, of the model apparently omitting the `PO`'s existing priorization, was also discovered.

The team finishes up the priority lane by speeding through a few work items in the "todo" state, confirming them to be still relevant but undone. Once the list is combed, it's time for the next board.

8.1.1.5 Outbox

The driver opens up *"OUTBOX (ska diskuteras snart)"* and commences the `OUTBOX` segment at **20:45** with a callout from the secretary. As per the Swedish language subtitle of the board, it is used to keep track of discussions that should take place.

`M1` immediately calls for a comment to be added on an existing work item that there's new evidence, referring to the `INTRO` segment discussion about a team being difficult to reach. This is apparently done for the benefit of a discussion to be had with `M2`, who is not present. Another discussion item is called out as being "longstanding", and is moved to that section of the board by the secretary.

Due to the excursion by `M1` of bringing a task in the middle of the board to the team's attention, the linear top-to-bottom fashion of reading the board that was seemingly faithfully observed before has broken down at this point, but is resumed. To-be-had discussions are gone through and confirmed relevant in much the same fashion as the todo items in the previous segment.

In the following excerpt there's again a reference to the difficulties discussed in the `INTRO`, now accompanied with a desire to not only have a discussion

but to *keep track* of whether a problem was in fact solved for good:

22:15 "'Tech team doesn't see documentation being handled.' Really don't!" (`M` quoting work item title)
"Well .." (`QA`)
"Again. This morning." (`M`)
"I have bumped and it's-" (`QA` stating a discussion was had)
22:25 "Yeah but until we can really say that it's now being properly handled ... we see a few cycles where it just .. goes well" (`M`)

Ostensibly this would mean that the discussion item will be held in the "longstanding" section for evaluation, even if such a discussion as intended by the item did in fact take place as claimed by `QA`.

Lastly, we observe a callback to the issue with fires and work in the module continuum. There's an unhandled item for which the discussion has already taken place, but the team would need to be informed about the outcome.

23:36 "'Ability to track module fires'" (`M` reading out item title as a prompt)
"Yeah, `PO` requested that-that any uhh module fires also be added to the module continuum so you can take a look at the module continuum afterwards and see which have been completed" (`D` relating the result of the discussion)
"Yes yes, very good" (`M`)

The PO's request itself happens to be an insignificant detail from the daily's perspective, because as shown in A1, the module continuum is never combed through in the daily. It provides another rationale for the tension observed earlier, however: the PO's wishes about work visualization in the continuum might have contributed to a conflict with the team's needs about transparency.

The "longstanding" discussion items are skipped with a callout from `M` get-

ting an affirmative response from the secretary. This concludes the `OUTBOX` segment.

8.1.1.6 Questions By this point, the team is evidently anxious to move forward: several members in succession call out the name of the next board, *"QUESTIONS"*. The purpose of the board is to make sure that information needs of parties, internal and customer alike, are not neglected. Concretely, there are both ongoing lines of inquiry and recurring "household keeping" themed tasks. Only the topmost tasks, belonging to the recurring category, are briefly confirmed as having been done.

24:39 "So ..." (`D1`)
24:41 "So ..." (`D2`)
24:43 "Let's not be done yet." (`M`)

Team members express desire to move on and perhaps finish the daily, but the manager wants to be more thorough. With that note, the driver shows the next board.

8.1.1.7 Non-functional continuum The non-functional continuum board consists of small tasks that have only superficial effect on the product and can thus be shipped immediately. The structure and sectioning is similar to "fires", but the work reflected therein is considered to be of less urgency by nature.

The manager appears to have had a clear agenda in moving to this board, because he proceeds to press on an uncompleted task.

24:48 "There's [item description]" (`M`)
"Has `D1`'s face on it [..]" (`QA`)
25:02 "So `D1` could you add these things? There's a huge organisational

waste when we have done [website content] already .. and they are not linked in the UI" (`M`)

25:12 "Yeah yeah" (`D1`)

"This is almost becoming a fire" (`M`)

"... Yeah." (`D1`)

25:21 "But like, we had this discussion before I left." (`M`)

"Yeah yeah I can add" (`D1`)

"But .. really" (`M`)

"Yeah yeah yeah yeah yeah" (`D1`)

25:29 "And... Start using the 'my tasks' thing [..]" (`M`)

"That's how I do it cause otherwise I just don't remember what I ..." (`D2`)

From the manager's perspective, the team appears to have failed in accounting for the timeliness requirement of the task being discussed. Proximally, the reason is pinned on `D1` not using the team's common tooling for supporting this by seeing that the task exists and is assigned. However, the team members seem to acknowledge that it is not trivial to make good priorizations of when these individual tasks would best be done, as they are unrelated to the team's larger deliverables.

8.1.1.8 Board overview

The `NON-FUNCTIONAL CONTINUUM` segment fluidly transforms into `BOARD OVERVIEW` when the manager again prompts:

26:05 "So what's the status of these numbered things? `QA`?" (`M`, referring to ongoing cycles)

No action is needed from the driver as the discussion items of this section are continuously visible on screen. `QA` elaborates on the statuses from their perspective. A developer is curious about the color coding `QA` uses for signifying project statuses, and gets an explanation. This is immediately taken into use by another developer by describing the state of the next project

on the list as "green", meaning "in progress".

26:50 "What's the definition of 'API'?" (`M` referring to cycle name)
"It was more that I didn't want to use the 'partner integration funnel' for my tasks-" (`D`)

27:01 "Okay.. could this be renamed so that it still means something? Like ... put in parenthesis that 'partner integration funnel'? Or something, because nobody has idea what's 'API' [in] this context" (`M`)

Manager calls for cycles being more clearly defined bundles of tasks, because a vague umbrella topic does not aid in understanding the work. The developer admits to not having been wanting to do this before. Perhaps the more specific name does not entirely describe the referred cycle's scope, or a better understanding has only recently arisen and the project's name is lagging behind.

Also of note is that the developer expresses having a sense of ownership over "his" tasks. There's apparently a domain that the developer personally is responsible for and this cycle would lie within that domain.

28:15 "Alright ... Now we're done." (`M`)

Nobody objects; the daily is concluded.

8.1.2 Relation between daily structure and kanban boards

Segments	Kanban board titles
INTRO	-
INBOX	INBOX (Composer 2)
FIRES	[FIRE]: This must be emptied
PRIORITY LANE	Priority lane

OUTBOX	OUTBOX (ska diskuteras snart)
QUESTIONS	QUESTIONS
NON-FUNCTIONAL CONTINUUM	CONTINUUM: Non-functional
BOARD OVERVIEW	-

Table III.5: List of daily segments and the kanban board list for daily `03` overlaid. The team proceeds through the boards as they are presented.

Chapter IV

Conclusions

9 Results

9.1 The daily reflects the activity system

9.2 The daily can be used as a tool for continuous improvement

9.3 The practice of a daily can inform software development activities

10 Practical implications

11 Evaluation

11.1 Limitations of the study

Appendix A

Dailies

A1 Daily segment lengths

Table A.1: Identified segments in dailies by their timestamp in the recording and duration.

Daily index	Segment	Timestamp	Duration
1	INTRO	00:02:02	00:12:28
	INBOX	00:14:30	00:13:30
	FIRES	00:28:00	00:04:40
	PRIORITY LANE	00:32:40	00:01:20
	OUTBOX	00:34:00	00:00:20
	QUESTIONS	00:34:20	00:01:43
	DONE	00:36:03	00:00:33
	RECORDING ENDS	00:36:36	
2	INTRO	00:01:11	00:08:59
	NON-CORE MODULE CONTINUUM	00:10:10	00:02:30
	CYCLES	00:12:40	00:03:00

Daily index	Segment	Timestamp	Duration
	INBOX	00:15:40	00:13:20
	FIRES	00:29:00	00:13:55
	PRIORITY LANE	00:42:55	00:02:30
	OUTBOX	00:45:25	00:01:25
	QUESTIONS	00:46:50	00:01:40
	DONE	00:48:30	00:00:23
	RECORDING ENDS	00:48:53	
3	INTRO	00:00:40	00:10:45
	INBOX	00:11:25	00:01:45
	FIRES	00:13:10	00:04:05
	PRIORITY LANE	00:17:15	00:03:30
	OUTBOX	00:20:45	00:03:20
	QUESTIONS	00:24:05	00:02:00
	"""BOARD OVERVIEW"""?"	00:26:05	00:02:13
	DONE	00:28:18	00:00:12
	RECORDING ENDS	00:28:30	
4	INTRO	00:00:00	00:04:40
	INBOX	00:04:40	00:12:10
	FIRES	00:16:50	00:09:40
	OUTBOX	00:26:30	00:03:20
	QUESTIONS	00:29:50	00:00:50
	NON-FUNCTIONAL CONTINUUM	00:30:40	00:01:45
	DONE	00:32:25	00:01:26
	RECORDING ENDS	00:33:51	
5	INTRO	00:00:19	00:07:56
	INBOX	00:08:15	00:07:25
	FIRES	00:15:40	00:05:50
	PRIORITY LANE	00:21:30	00:00:20

Daily index	Segment	Timestamp	Duration
	OUTBOX	00:21:50	00:02:20
	QUESTIONS	00:24:10	00:03:43
	DONE	00:27:53	00:00:23
	RECORDING ENDS	00:28:16	
6	INBOX	00:00:07	00:07:23
	FIRES	00:07:30	00:04:50
	PRIORITY LANE	00:12:20	00:01:40
	OUTBOX	00:14:00	00:01:10
	QUESTIONS	00:15:10	00:00:20
	DONE	00:15:30	00:02:17
	RECORDING ENDS	00:17:47	
7	INTRO	00:00:13	00:03:17
	INBOX	00:03:30	00:13:00
	FIRES	00:16:30	00:05:10
	PRIORITY LANE	00:21:40	00:01:05
	OUTBOX	00:22:45	00:00:55
	QUESTIONS	00:23:40	00:01:45
	NON-FUNCTIONAL CONTINUUM	00:25:25	00:01:05
	MODULES CONTINUUM	00:26:30	00:00:25
	DONE	00:26:55	00:00:13
	RECORDING ENDS	00:27:08	
8	INTRO	00:00:20	00:06:25
	INBOX	00:06:45	00:07:00
	FIRES	00:13:45	00:02:15
	PRIORITY LANE	00:16:00	00:02:00
	QUESTIONS	00:18:00	00:00:30
	NON-FUNCTIONAL CONTINUUM	00:18:30	00:01:40
	DONE	00:20:10	00:00:16

Daily index	Segment	Timestamp	Duration
	RECORDING ENDS	00:20:26	
9	INTRO	00:00:30	00:01:40
	INBOX	00:02:10	00:05:20
	FIRES	00:07:30	00:03:05
	OUTBOX	00:10:35	00:01:10
	QUESTIONS	00:11:45	00:02:15
	NON-FUNCTIONAL CONTINUUM	00:14:00	00:01:10
	BOARD OVERVIEW?	00:15:10	00:02:40
	DONE	00:17:50	00:01:13
	RECORDING ENDS	00:19:03	
10	INTRO	00:00:25	00:05:55
	INBOX	00:06:20	00:05:39
	FIRES	00:11:59	00:10:11
	PRIORITY LANE	00:22:10	00:02:35
	OUTBOX	00:24:45	00:05:15
	QUESTIONS	00:30:00	00:00:35
	DONE	00:30:35	00:00:38
	RECORDING ENDS	00:31:13	
11	INBOX	00:00:15	00:05:30
	FIRES	00:05:45	00:02:00
	PRIORITY LANE	00:07:45	00:00:30
	OUTBOX	00:08:15	00:00:10
	QUESTIONS	00:08:25	00:02:57
	DONE	00:11:22	00:00:35
	RECORDING ENDS	00:11:57	
12	INTRO	00:00:08	00:07:52
	INBOX	00:08:00	00:03:20

Daily index	Segment	Timestamp	Duration
	FIRES	00:11:20	00:09:36
	PRIORITY LANE	00:20:56	00:01:29
	OUTBOX	00:22:25	00:09:35
	QUESTIONS	00:32:00	00:00:22
	DONE	00:32:22	00:02:06
	RECORDING ENDS	00:34:28	
13	INTRO	00:00:58	00:00:47
	BOARD OVERVIEW	00:01:45	00:15:45
	INBOX	00:17:30	00:20:18
	FIRES	00:37:48	00:05:32
	PRIORITY LANE	00:43:20	00:00:28
	OUTBOX	00:43:48	00:00:32
	QUESTIONS	00:44:20	00:00:20
	NON-FUNCTIONAL CONTINUUM	00:44:40	00:00:35
	NON-CORE MODULE CONTINUUM	00:45:15	00:01:35
	DONE	00:46:50	00:00:43
	RECORDING ENDS	00:47:33	
14	INTRO	00:00:26	00:02:07
	INBOX	00:02:33	00:18:52
	FIRES	00:21:25	00:04:15
	PRIORITY LANE	00:25:40	00:00:35
	OUTBOX	00:26:15	00:06:15
	QUESTIONS	00:32:30	00:00:45
	NON-FUNCTIONAL CONTINUUM	00:33:15	00:01:45
	NON-CORE MODULE CONTINUUM	00:35:00	00:00:42
	DONE	00:35:42	00:00:24
	RECORDING ENDS	00:36:06	
15	INTRO	00:00:25	00:01:35

Daily index	Segment	Timestamp	Duration
	INBOX	00:02:00	00:05:55
	NON-FUNCTIONAL CONTINUU	00:07:55	00:05:05
	INBOX AGAIN	00:13:00	00:14:55
	FIRES	00:27:55	00:06:40
	PRIORITY LANE	00:34:35	00:02:40
	OUTBOX	00:37:15	00:08:05
	QUESTIONS	00:45:20	00:02:25
	NON-FUNCTIONAL CONTINUUM	00:47:45	00:01:15
	NON-CORE CONTINUUM	00:49:00	00:02:28
	DONE	00:51:28	00:00:16
	RECORDING ENDS	00:51:44	
16	INTRO	00:00:12	00:12:33
	OUTBOX	00:12:45	00:20:05
	PRIORITY LANE	00:32:50	00:00:30
	OUTBOX	00:33:20	00:07:13
	INBOX	00:40:33	00:15:30
	FIRES	00:56:03	00:03:45
	DONE	00:59:48	
	RECORDING ENDS	01:00:21	
17	INTRO	00:01:11	00:01:54
	FIRES	00:03:05	00:12:47
	INBOX	00:15:52	00:22:08
	FIRES	00:38:00	00:00:25
	PRIORITY LANE	00:38:25	00:01:05
	OUTBOX	00:39:30	00:04:00
	MONEYBOX	00:43:30	00:00:45
	QUESTIONS	00:44:15	00:00:20
	NON-FUNCTIONAL CONTINUUM	00:44:35	00:05:45
	NON-CORE MODULES	00:50:20	00:02:53

Daily index	Segment	Timestamp	Duration
	DONE	00:53:13	00:00:22
	RECORDING ENDS	00:53:35	
18	INTRO	00:00:15	00:03:20
	CYCLES	00:03:35	00:12:20
	INBOX	00:15:55	00:12:50
	FIRES	00:28:45	00:06:20
	PRIORITY LANE	00:35:05	00:00:15
	OUTBOX	00:35:20	00:02:25
	QUESTIONS	00:37:45	00:00:40
	NON-FUNCTIONAL	00:38:25	00:00:25
	NON-CORE MODULES	00:38:50	00:01:25
	DONE	00:40:15	00:00:14
	RECORDING ENDS	00:40:29	
19	INTRO	00:01:15	00:09:55
	INBOX	00:11:10	00:20:20
	FIRES	00:31:30	00:12:45
	OUTBOX	00:44:15	00:01:18
	MONEYBOX	00:45:33	00:00:04
	WATBOX	00:45:37	00:00:13
	QUESTIONS	00:45:50	00:00:30
	MARKETING BOX	00:46:20	00:00:13
	NON-CORE MODULE CONTINUUM	00:46:33	00:00:57
	NON-FUNCTIONAL CONTINUUM	00:47:30	00:02:10
	DONE	00:49:40	00:00:17
	RECORDING ENDS	00:49:57	
20	INTRO	00:00:16	00:04:04
	INBOX	00:04:20	00:18:25
	FIRES	00:22:45	00:10:55

Daily index	Segment	Timestamp	Duration
	PRIORITY LANE	00:33:40	00:01:45
	OUTBOX	00:35:25	00:02:35
	NON-FUNCTIONAL CONTINUUM	00:38:00	00:00:45
	NON-CORE MODULE CONTINUUM	00:38:45	00:00:55
	DONE	00:39:40	00:01:31
	RECORDING ENDS	00:41:11	

Appendix B

Kanban boards

B2 Lists of kanban boards

Table B.1: Partial outlines of the kanban boards used by the team ordered by date.

Capture date	Boards
2015-12-08	INBOX (Composer 2) [FIRE]: This must be emptied Priority lane OUTBOX (ska diskuteras snart) QUESTIONS CONTINUUM: Modules CONTINUUM: Non-functional (omitted)
2015-12-16	INBOX (Composer 2) [FIRE]: This must be emptied Priority lane

Capture date	Boards
	OUTBOX (ska diskuteras snart)
	QUESTIONS
	CONTINUUM: Non-functional
	CONTINUUM: Modules
	#15c: iOS 4.0.8
	#17 Android 4.2.x
	#19 Composer 2
	#20 Partner API
	#20 Partner integration funnel
	#20 Release Wizard
	#21 Logic editor
	#?? DreamFactory Funnel
	#?? Composer 2
	#?? iOS
	#?? Android
	#?? Payments
	#?? Dolan
	#2X API and Friends
	(omitted)
2016-01-18	INBOX (Composer 2)
	[FIRE]: This must be emptied
	Priority lane
	OUTBOX (ska diskuteras snart)
	MONEYBOX
	QUESTIONS
	MARKETING
	CONTINUUM: Non-functional
	CONTINUUM: Modules
	#17 Android 4.2.0 + 4.2.1
	#22 Composer 2

Capture date	Boards
	#23 Release Lifecycle
	#23 Composer 2
	#23 API and ACL
	#24 Release Lifecycle
	#24 Logic editor
	#24 Composer 2
	#?? iOS 4.1.0
	#?? Android 4.3
	#?? Payments
	#?? Dolan
	#?? Module core
	#?? API and friends
	#?? Push Notifications
	#?? SandboxDB 2
	INBOX (Kisko)
	Exception Box
	—
	#NN Descriptive Theme Name
	#NN Release Lifecycle
	—
	BACKLOG: Modules
	—
	MINOR Roadmap
	SOLUTION Roadmap
	Composer 2 bug backlog
	Toolchain bug backlog
	SORT THIS AWAY PLS
	NEXT
	SOON
	LATER
	WAT

Capture date	Boards
	HARD AND SCARY
	WATBOX
	TECH WISHING WELL
	(omitted)
2016-01-19	(omitted)
	INBOX (Kisko)
	MONEYBOX
	MARKETING
	TECH WISHING WELL BOX
	WATBOX
	EXCEPTION BOX
	BACKLOG: Modules
	(omitted)

B3 Lists of kanban board structures

Table B.2: Partial outlines of the structures of the kanban boards used by the team ordered by date of capture.

Capture date	Board name	Labels
2015-12-16	CONTINUUM: Modules	(unlabeled)
		!! NOT AN INBOX !!
		SMOKETEST FAILED (in production)
		DEPLOYMENTS (make sure all subtasks ar
		WAITING FOR DEVQA (in devgyver)
		DEVQA REJECTED (in devgyver)
		WAITING FOR MERGE (in branch)
2015-12-16	[FIRE]: This must be emptied	(unlabeled)

Capture date	Board name	Labels
		UNHANDLED FIRES INBOX
		AWAITING SMOKE TEST (in production)
		SMOKETEST FAILED
		DEPLOYMENTS (make sure all subtasks ar
		WAITING IN DEVGYVER FOR MERGE
		IMPEDED
		IN DEV QA
		BACKLOG (waiting to be picked on the tab
		WAITING FOR REPRO
		LONGSTANDING (extinguished on its own
2015-12-16	INBOX (Composer 2)	(unlabeled)
		INBOX TUTORIAL
		Inbox
		Needs something
		Discuss later
		Wishlist inbox
		Wishlist
		(rest omitted)
2015-12-16	OUTBOX (ska diskuteras snart)	(unlabeled)
		DONE
		TODO
		IMPEDED
		Longstanding (WIP limit: 18)
2015-12-16	Priority lane	(unlabeled)
		DONE
		DOING
		TODO
		LONGSTANDING

Capture date	Board name	Labels
2015-12-16	QUESTIONS	(unlabeled) Inbox Updates asked for Seen Badly needs a repro Client Plugins Misc CLI Needs repro
2015-12-21	[FIRE] This must be emptied	(omitted) WAITING IN DEVGYVER FOR MERGE IMPEDED IN DEV QA UNICORN BACKLOG (wip: 1) JUNIPER BACKLOG (wip: 1+1) BACKLOG (wip: infinite) WAITING FOR REPRO LONGSTANDING (extinguished on its own
2015-12-21	OUTBOX (ska diskuteras snart)	(unlabeled) INBOX ABOVE DONE ALMOST-BUT-NOT-QUITE-DONE TODO IMPEDED Longstanding (WIP limit: 18)
2016-01-07	OUTBOX (ska diskuteras snart)	(unlabeled)

Capture date	Board name	Labels
		INBOX ABOVE DONE ALMOST-BUT-NOT-QUITE-DONE TODO IMPEDED MISSING IN ACTION
2016-01-12	MONEYBOX	(unlabeled) PENDING CUSTOMER ACTION
2016-01-12	OUTBOX (ska diskuteras snart)	(omitted) DONE ALMOST-BUT-NOT-QUITE-DONE TODO BOUNCED NEEDS PLANNING IMPEDED MISSING IN ACTION

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.

Silvia Gherardi. Organizational learning: The sociology of practice. *Handbook of Organizational Learning and Knowledge Management*, 2:43–65, 2011.

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.