Aalto University
School of Science
Degree Programme in Information Networks

Eevert Saukkokoski

What exactly is the use of dailies:

A practice-based perspective on continuous improvement in software development

Master's Thesis
Espoo, November 23rd, 2016

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

Aalto University
School of Science
Degree Programme in Information Networks

ABSTRACT OF
MASTER'S THESIS

Author:	Eevert Saukkokoski		
Title:	What exactly is the use of dailies: A practice-based perspective on continuous improvement in software development		
Date:	November 23rd, 2016	**Pages:**	130
Major:	Business processes and services in digital networks	**Code:**	TU-124
Supervisor:	Professor Riitta Smeds, D.Sc. (Tech)		
Advisor:	Otso Hannula, M.Sc. (Tech.)		

The daily meeting is a commonplace agile software development method from the Scrum methodology. Agile methods are described as adaptable, but as they are often products of trial-and-error, successful adaptation can be difficult. Lean software development supports continuous improvement and offers principles based on which suitable methods can be created ad hoc. The introduction of lean calls into question the use of methods which add processual overhead, such as the daily.

This thesis studies the use of a daily in support of software development activities. An understanding is developed on how continuous improvement in practices may be analysed through the lens of practice-based activity theory. Guidelines leveraging the perspective of activity theory are offered for the development of practices where continuous improvement is a goal.

The theoretical objective is to find a means of description for the mechanism of continuous improvement in the practice of software development. Agile and lean methodology literature are joined by a perspective on improvement as a social process of innovation from activity theory, whereby improvement can be identified as transformations in a shared object of activity.

This thesis presents an ethnographical case study conducted within a software development team using a lean-inspired virtual kanban workflow model in their daily meetings. The team is observed over 20 dailies and audio recordings combined with screenshots of the kanban model are used to analyze how social interactions in the daily support continuous improvement.

The results of the study suggest that a daily practice may contribute to continuous improvement by providing a structure in which a visualization of work such as kanban can be used collaboratively, allowing failures in the practice to be brought up as disturbances which can be used as motives for transforming the practice. By pointing out disturbances as necessary for social innovation and showing the role they play in continuous improvement efforts, this thesis outlines a method for finding bottlenecks to learning in lean software development.

Keywords:	software development methods, agile, daily, lean, kanban, continuous improvement, practice-based research, activity theory
Language:	English

Aalto-yliopisto
Perustieteiden korkeakoulu
Informaatioverkostojen koulutusohjelma

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä:	Eevert Saukkokoski		
Työn nimi:	Mitä hyötyä on daily-käytännöstä: Käytäntölähtöinen katsaus ohjelmistokehityksen jatkuvaan parantamiseen		
Päiväys:	23. marraskuuta 2016	**Sivumäärä:**	130
Pääaine:	Liiketoiminta- ja palveluprosessit tietoverkoissa	**Koodi:**	TU-124
Valvoja:	Professori Riitta Smeds, TkT		
Ohjaaja:	Otso Hannula, Tekn. DI		

Daily eli päivittäinen tapaaminen on ketterässä ohjelmistokehityksessä usein käytetty menetelmä, jonka alkuperä on Scrum-menetelmäkehyksessä. Ketteriä menetelmiä kuvaillaan helposti muokattaviksi, mutta koska menetelmät on useimmiten luotu yrityksen ja erehdyksen kautta, on niiden muokkaaminen onnistuneesti hankalaa. Lean-perusteinen ohjelmistokehitys tukee jatkuvaa parantamista ja tarjoaa periaatteita, joiden avulla voidaan luoda tarpeisiin muokattuja menetelmiä. Leanin käyttö asettaa kyseenalaiseksi daily-menetelmän kaltaiset, suoraan arvoa tuottamattomat prosessin osat.

Tämä tutkimus perehtyy daily-menetelmän käyttöön jatkuvan parantamisen tukena ohjelmistokehityksessä. Tutkimuksessa kehitetään ymmärrystä siitä miten käytäntöjen jatkuvaa parantamista voidaan analysoida käytäntölähtöisen toiminnan teorian kautta. Tutkimus esittää toiminnan teoriaan pohjautuvia ohjeita sellaisten käytäntöjen muodostamiseen, joissa tavoitteena on jatkuva parantaminen.

Tutkimuksen teoreettinen tavoite on löytää tapa kuvailla mekanismeja, joiden avulla ohjelmistokehityksen jatkuva parantaminen tapahtuu käytännössä. Ketterän ja lean-ohjelmistokehityksen kirjallisuuden näkökulmia täydennetään toiminnan teorialla. Siinä toiminnan parantaminen nähdään sosiaalisena innovointiprosessina, joka ilmenee yhteisen toiminnan kohteessa tapahtuvina muutoksina.

Empiirisenä materiaalina esitetään etnografinen tapaustutkimus ryhmästä, joka käyttää daily-tapaamistensa yhteydessä lean-perusteista virtuaalista kanban-mallinnusta työnsä ohjaukseen. Ryhmää tarkkaillaan 20 daily-tapaamisen ajan, ja tarkkailujaksolta kerättyjä äänitallenteita sekä kuvakaappauksia kanban-mallista käytetään analysoimaan jatkuvaan parantamiseen johtaneita interaktioita.

Tulosten perusteella daily-menetelmä tukee jatkuvaa parantamista tarjomalla rakenteen, jossa kanbanin kaltaisia visualisaatioita voidaan käyttää yhteisesti. Mahdolliset virheet nousevat esiin häiriöinä, joita voidaan hyödyntää motiiveina käytännön muuttamiselle. Tutkimus tarjoaa tavan löytää esteitä lean-ohjelmistokehityksessä tapahtuvalle oppimiselle näyttämällä häiriöiden hyödyntämisen olevan välttämättömiä sosiaaliselle innovaatiolle.

Asiasanat:	ohjelmistokehitysmenetelmät, ketteryys, daily, lean, kanban, jatkuva parantaminen, käytäntölähtöinen tutkimus, toiminnan teoria
Kieli:	Englanti

Acknowledgements

This thesis would not have been without

Reaktor, who set me on my journey to understanding lean.
Futurice, who contributed a copy of Lean Software Development.
AppGyver, who served as inspiration.
Venuu, who showed flexibility, compassion and empathy.
Otso, who provided extraordinary support.
Anna, my head facilitator for life.
D.

Helsinki, November 23rd, 2016

Eevert Saukkokoski

Contents

List of Tables

List of Figures

Chapter I

Introduction

> *We want our organizations to be adaptive, flexible, self-renewing, resilient, learning, intelligent – attributes found only in living systems. The tension of our times is that we want our organizations to behave as living systems, but we only know how to treat them as machines.*
>
> (Wheatley and Kellner-Rogers, 1996, The irresistible future of organizing)

This thesis studies the use of a daily meeting in support of software development activities. The study searches for evidence that the daily contributes to a team's capability for innovation. An ethnographical case study conducted within a software development team using a kanban work visualization in their daily meetings is described. The case study is analyzed through the lens of practice-based activity theory to identify how the daily enables continual transformation of the kanban system.

The theoretical objective of this thesis is to describe the social mechanisms of continuous improvement in the practice of software development. Continuous

improvement is taken as the defining characteristic of methods that aim at waste reduction according to lean principles, such as the use of a kanban system. The practical contribution of the analysis is to efforts of developing methods and practices for supporting software development in scenarios where continuous improvement is desired.

This chapter describes the background against which this study is set and delineates the research problem. Theoretical research questions are posed based on the research problem, to be answered in chapter II. Finally, the scope and goals of the study are defined and the structure of this thesis is introduced.

1 Background and motivation

The daily has become part of IT industry vernacular. The word refers to a meeting of software development team members taking place every day (Sutherland and Sutherland, 2014). Stemming from the popular *Scrum* methodology of *agile software development*, dailies are a very commonplace occurrence in the field (West et al., 2010). Agile encompasses methodologies which reflect the quality of "agility" to varying degrees (Abrahamsson et al., 2002) as well as a manifesto which encapsulates some of its early evangelists' most important values (Beck et al., 2001). Methodologies under the agile umbrella have been characterized with simplicity and ease of adaptation as key (Abrahamsson et al., 2002). When agile methods such as the daily are applied, they don't always work as introduced (Marchenko and Abrahamsson, 2008). The situation is problematic for practitioners, because successfully modifying methods that have originally been developed by trial-and-error is difficult (Poppendieck and Poppendieck, 2003, p. xiii-xvi).

A new school of *lean software development* aspires to supplant Scrum and its ilk as the *de facto* approach to software processes (Ahmad et al., 2013). Lean's

purported advantage is that it provides principles based on which concrete methods may be developed (Poppendieck and Poppendieck, 2003). The perspective lean offers on software development is that it can be seen as a value-producing system. This systems thinking is evident in lean-derived methods such as kanban, with its focus on visualizing and managing flow of work (Ahmad et al., 2013). Lean aims at process improvement through reduction of waste, or non-value-adding activities (Poppendieck and Poppendieck, 2003). More broadly, lean-derived methods can be described as striving towards *continuous improvement* (Bhuiyan and Baghel, 2005).

In agile literature, the daily is posited as an opportunity for face-to-face interaction, the necessity of which is held to be critical for transfer of ideas and achieving innovative results (Highsmith and Cockburn, 2002). The introduction of lean calls into question the use of the agile daily, which must be motivated not only through its role in agile methodology but by its usefulness. Ineffective use of practitioners' time is one of the criticisms aimed at agile implementations (Begel and Nagappan, 2007). Such criticism is especially relevant when directed against Scrum methods, as the methodology is focused on project management (Marchenko and Abrahamsson, 2008) – that is, activities which are not value-adding *per se*.

Lean's goal of continuous improvement and agile's claim of innovation in face-to-face interaction have parallels which intersect in practice. It cannot be that the mere existence of a kanban modelling, for instance, allows continuous improvement to take place. Instead, it must be that the improvement happens somehow in conjunction with the use of such a method by practitioners. Likewise, investigating the assertion that dailies enable innovation in social interaction requires that we look into what happens in practice. The tools of neither agile nor lean are apt at describing such mechanisms, so we must reach for understanding elsewhere.

Innovation in social interactions has been studied in practice-based *activity theory* (Engeström, 1987). The *practice-based approach* of organizational

research looks at knowledge as acquired, used and observed 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). In this thesis, a case study of software engineering practice is described and activity theory is used as a lens through which to observe continuous improvement during daily activities. The objective is to support the development and evaluation of software development activities which aim at continuous improvement and motivate the use of a daily as one such device.

2 Research problem and research questions

Based on section 1, the context of the problem is as follows. Agile finds that innovation happens in face-to-face interactions, with dailies being a specific example. Lean claims continuous improvement of a production system is supported with the use of lean-derived methods, such as kanban. The practice based approach sees an equivalence between practicing, knowing and learning.

If dailies support such system-level innovation as is meant by continuous improvement, this quality should be observable in daily activity. What kind of learning can we observe in the daily, and what enables it? A way to describe the mechanism of continuous improvement through the practice based perspective is needed. The research problem for this thesis is:

RP: How does a daily support continuous improvement?

The research problem leads us to a threefold investigation of literature in order to understand the fundamentals in play.

TRQ1: How does a daily in agile software development support the development team?

TRQ2: How does the use of a kanban system support continuous improvement in the context of lean software development?

TRQ3: How can continuous improvement be observed in practice?

Question *TRQ1* aims at setting a point of comparison which we can take to reflect on the qualities of the daily that we will be observing in the empirical study. With *TRQ2* we probe the role of kanban systems within the context of lean and relate it to the lean principle of continuous improvement. We find a language for describing the social process we're observing in *TRQ3* by looking beyond software development methodology literature and delving into the practice-based approach.

3 Research approach, scope and methods

The goal of the study is broadly to understand a specific practice situated within the field of software development. This premise necessitates a qualitative approach to analysis (Jonassen and Land, 2012). Lethbridge et al. (2005) claim that to truly understand the process of creating software, it is essential to study it in real environments. Perhaps owing to this fact, even while qualitative studies of software practice appear relatively rare (Robinson et al., 2007), the agile discourse has a history with ethnographically-inspired studies of agile practices (Robinson et al., 2007; Marchenko and Abrahamsson, 2008).

LeCompte and Goetz (1982) have characterised the ethnographical approach with emphasising observation, focusing on natural settings, using participant

constructs for structuring research, and avoidance of manipulation of study variables by investigators. Robinson et al. (2007) add that the ethnographically inclined researcher attempts to understand practice on its own terms, studying the totality of it with all its messy characteristics, and aspires not to disturb the practice. Ethnography is suited for situations where unique processes of change may be observed (LeCompte and Goetz, 1982), which also implies that the situation or practice should be observed over a period of time (Jonassen and Land, 2012). The research strategy that focuses on understanding dynamics present within a single setting is called case study, which may be conducted over single or multiple cases (Eisenhardt, 1989).

The research method in this study is an ethnographically-inspired single case study. This means that rich data was collected by the researcher as part of the community of practice within a single setting in order to provide rich account of the practice (Robinson et al., 2007). The aim was to eventually find theory to explain the data (LeCompte and Goetz, 1982). This thesis applies abductive reasoning, or inference to the best explanation (Ketokivi and Mantere, 2010), in the process of which the theoretical framework and the empirical analyses inform one another (Dubois and Gadde, 2002). Field studies easily generate vast amounts of data (Lethbridge et al., 2005), and not all of it is practical to analyse; when the researcher simultaneously collects, codes and analyses data in order to decide on the next object of inquiry by means data, this is called theoretical sampling (Coyne, 1997). The nonlinear, iterative movement between theory and empirical data is called *systematic combining* by Dubois and Gadde (2002) in their overview of abductive approaches on case research.

The context of the study is an organisation where the software development team applies both daily meetings and a kanban system in their work. The object of study is how the overlapping agile and lean practices function together and how their use might support the development team. The case is described further in section 9. Data was collected primarily as audio

recordings from daily meetings, but it was amended with screenshots from different aspects of the kanban system and field notes, with the latter suggested for speeding up analyses by Eisenhardt (1989).

Lethbridge et al. (2005) describe three degrees of data collection techniques suitable for field studies of software engineering. First degree techniques are observational and provide a real-time view of the phenomena, but yield material that is difficult to analyse due to both the material's density and the knowledge required to interpret it correctly. In second degree techniques, participants are studied in their environment but without the researcher being present simultaneously while the data is collected. Not necessitating direct contact between researcher and participants makes these techniques applicable for longitudinal studies. Techniques of the third degree attempt to gain information by inspecting artefacts produced by workers, which are often easily accessible and require no commitment from participants but are removed from the actual process being observed. Lethbridge et al. (2005) note that material in the form of artefacts removed from the context of the action being studied is difficult to interpret successfully, and so recommend third degree techniques to be supplemented by other ones.

Lethbridge et al. (2005) define the two main data collection techniques relevant to this study. Having the researcher take part in the observed practice is a first degree technique named *participant observation*. Capturing the state of the kanban system on the other hand constitutes a third degree technique called *analysis of electronic databases of work performed*. Lethbridge et al. (2005) place possible means of record-keeping on a spectrum according to dimensions of intrusiveness and time-intensivity. Videotape would be most complete but also most intrusive and time-intensive, while manual record keeping would be the least complete and intensive. Audio is situated in the middle of the spectrum due to being fairly complete but losing details of the physical environment. Crucial features for this study were considered to be low intrusivity for participants and sufficiently low time-intensivity with

regards to analysis. The use of audio recordings as a primary source of data was found to be a reasonable compromise. The primary data also aptly serves as means of contextualizing and interpreting the secondary data captured by third degree techniques.

In analysing learning situations, Jonassen and Land (2012) consider it important to understand the overall direction of the activity. They suggest that practice researchers first study broad patterns of activity before considering narrow episodic fragments. The large bodies of data produced in field studies must also be reduced to a more comprehensible format in a process of coding used to categorize the data according to a schema, which can be high level but must be guided by the goals of the research (Lethbridge et al., 2005). The analysis in this study progressed in the following structure, starting out with preliminary analytic 'preconceptions' (Dubois and Gadde, 2002) and refining understanding in iterative rounds:

1. Overview of data based on field notes and audio, creation of a daily content log and outline of activities. Based on preconceptions, attention was paid on fragments that stood out as unusual.
2. A structure for the dailies was identified using the team's own constructs as a basis, with secondary and primary data corroborating each other.
3. Segments in the content log were coded according to the identified structure and catalogued for analysis.
4. Episodes of particular interest were reviewed from the audio recordings with the aid of the content log based on newly gained theoretical understanding.
5. A subset of episodes was further sampled such that episodes from different dailies could be linked thematically.

In summary, the research follows an abductive process of inference to the best explanation in search of a theory that would explain the data, as is typical of ethnographical research. Such research has been done on agile

practices before, and the approach is applicable for looking into processes of change and learning. The research studies that which occurs naturally and attempts not to control the research context. This means that instead of testing hypotheses on the efficacy of different measures, we attempt to provide a rich, translatable account of the findings and process of reasoning such that the applicability of the theoretical constructs used may be evaluated by the reader.

4 Structure of the study

The study is presented in four chapters: introduction, theoretical framework, empirical study, and conclusions. Table 1 illustrates the line of argumentation throughout the study.

I. Introduction	Background and motivation
	Research problem
II. Theoretical framework	Agile software development; dailies
	Lean software development; kanban
	Activity theory; improvement as innovation
	Theoretical synthesis and research questions
III. Empirical study	Empirical study description
	Data collection
	Findings on dailies
IV. Conclusions	Research results
	Study implications
	Evaluation

Table 1: Structure of the study.

In chapter I, the background and motivation of the study is presented. The introduction is completed by a description on the research problem undertaken, the approach chosen and the methods of the study.

Chapter II, theoretical framework, encompasses a review of relevant literature for purposes of understanding the problem domain and allowing for analysis

of the empirical data. The background and contents of agile software development is discussed, they are linked to lean software development, and finally a complementary, practice-based viewpoint is taken through activity theory. The theoretical synthesis on these perspectives allows us to formulate the questions for the empirical study.

Chapter III describes a case study, its data gathering and analysis. Findings from the study are presented to allow for answering the empirical research questions.

In chapter IV, the empirical research questions are answered in light of the theoretical understanding gained from chapter II. The results are combined to formulate an answer to the research problem. Finally, the implications of the result are discussed and the study is evaluated.

Chapter II

Theoretical framework

This chapter contains a study of relevant literature on agile software development, lean software development, and the practice based perspective on innovation. The chapter is structured to answer the theoretical research questions posed in chapter I.

Section 5 discusses the background of agile, an overview of the Scrum methodology, and the daily as defined in Scrum. Section 6 relates agile to lean and defines the role of continuous improvement and kanban. Section 7 investigates the social process of innovation from the practice based perspective and suggests an activity theoretical basis for empirical analysis. A theoretical synthesis is provided in section 8.

5 Dailies in Agile

This section relates the background of how the agile movement, the Scrum methodology and the daily method came to be. Basic principles of Scrum are introduced and the rules of a daily are defined.

5.1 Discovering Scrum

Common wisdom used to hold that software development could be modelled after a *plan-driven* fashion (Abrahamsson et al., 2002). These models are well-researched and mature (Huo et al., 2004), but heavy. They assume that requirements may be discovered beforehand, that they will not undergo significant change through the duration of a project and that, fundamentally, the process of developing software is predictable and repeatable (Sutherland, 2001). The quintessential example of a plan-driven methodology[1] would be the *waterfall*[2]: gather requirements, devise a solution, program the solution, bring it to customers (Sutherland, 2001; Huo et al., 2004). Rinse and repeat.

Turns out, this analytical view of software processes has multiple points of failure. To start with: a project's premises may change on the way (Highsmith and Cockburn, 2002) and defining requirements is notoriously tricky (Lindstrom and Jeffries, 2004). In fact it appears that plan-driven models on the whole don't reflect the reality of software development well at all – especially in fast growing industries (Abrahamsson et al., 2002), where every company is trying to get the jump on their competitors.

Enter agility. The *Manifesto for Agile Software Development* (Beck et al., 2001) introduced the umbrella moniker "agile" as a word to describe a new class of models. Reality being complex and diverse, it's no surprise that the manifesto itself covers five different models (Fowler and Highsmith, 2001), and *e.g.* Abrahamsson et al. (2002) identify eight. Agility may well be seen as an ideological movement, or a culture of which many interpretations exist (Glass, 2001; Kruchten, 2007). Agile models have been characterised by Abrahamsson

[1] *Methodology* is an ascriptive description of techniques: what to do, how, when, by whom and so on.

[2] It should be noted that this simplistic construction of the waterfall is essentially a strawman argument. Winston Royce's 1970 original work detailing the "waterfall" itself remarked how a process without feedback cycles was unlikely to work. (Poppendieck and Poppendieck, 2003, p. 24-25)

et al. (2002) with the following attributes:

Incremental Delivery of software in short cycles

Cooperative Developers and their clients in unhindered, continuous communication

Straightforward Easy to learn, well documented, readily adaptable

Adaptive Options are not locked down, making alterations as needed is enabled

The term agile is thus identified as remarkably polysemous and dependant on context. To get more concrete, we will be discussing a specific agile process methodology called *Scrum*.

5.2 Overview on the Scrum methodology

Scrum is part of the original suite of methodologies giving rise to the agile manifesto (Beck et al., 2001). It has also been recognized among the most popular methodologies (West et al., 2010) and heralded as virtually a de-facto industry standard (Marchenko and Abrahamsson, 2008). Albeit no single method is endorsed by all things agile (Kruchten, 2007), in terms of industry adoption and influence on generic expectations of what it means to be agile, there is simply no better example than Scrum.

Schwaber (1995) defines the Scrum methodology as a process of software development consisting of phases of pregame, game and postgame[3]. Planning, conceptualization, analysis, and high level architectural design are part of

[3]The curious use of the term *game* stems from a sports analogy by Takeuchi and Nonaka (1986). They argued that software development is not best seen as a game of passing the baton, but as a rugby team advancing the ball by passing it back and forth between team members. It is not coincidental that Schwaber's use of the word Scrum to describe his methodology also derives from rugby.

pregame. Postgame consists of the project's closure, when the product under development is deemed ready and will be released. Game is where the product is developed in iterative increments called sprints. It is also the part where Schwaber distinguishes the methodology from the derided waterfall; without the keyword *iterative*, one could find a very obvious mapping from the Scrum words to the world of waterfall.

A sprint is defined by Schwaber (1995) as a collaborative team performance over a preset period of time, where the processes of develop, wrap, review and adjust take place. Development "consists of the micro process of discovery, invention and implementation". Wrapping is about making sure the work is observable, eg. that the resulting software can be run. Review means a meeting where progress is presented, problems raised and resolved, identified risks observed and responded to. Adjustment refers to taking the freshly gained information and consolidating it to shared artefacts.

Work items that describe how the product fails to satisfy current requirements are dubbed the backlog (Schwaber, 1995), from which the most important ones are prioritised for completion in a sprint in a sprint planning event (Schwaber and Sutherland, 2011). After a sprint has ran to completion, it is followed by an all-hands sprint review (Schwaber, 1995). This represents an opportunity for reflecting on what was learned in the last sprint and reprioritizing or modifying the work items accordingly (Highsmith and Cockburn, 2002).

Here we've seen an outline of Scrum and how it applies to the whole process of software development. So far our level of abstraction is at the *sprint* stage, or weeks of calendar time. But what about the day-to-day work? In the next section we will see how this is supported in Scrum by the daily Scrum meeting.

5.3 The archetypal Scrum daily

> The heart of the Scrum is the daily stand-up meeting (West et al., 2010)

The original Scrum formulation by Schwaber (1995) does not explicate the need of a daily. Scrum's co-originator (Beck et al., 2001), Jeff Sutherland, describes in his book on Scrum (Sutherland and Sutherland, 2014) that he was inspired in 1993 by a kind of warrior ceremony of the Maori people from New Zealand called the 'haka':

> The haka is a warrior dance that charges up people about to go into battle. While watching it, you can almost see the energy come out of each player and coalesce into a greater whole. With synchronized stomping and clapping and chanting—ritualized movements of cutting an enemy's throat—you can see ordinary men transform themselves into something bigger, something greater. They're invoking a warrior spirit that does not accept defeat or dismay.

Sutherland's team analysed the practices of successful teams and found an example in Borland Software Corporation, who gathered their team together every single day to discuss progress and resolve challenges. In trying to find a way to distill Borland's one-hour daily to its essence, Sutherland with his team ended up with the idea of *three questions* and a set of rules.

According to the rules, the daily:

1. Should be held at the same time every day and be attended by everyone.
2. Should not last more than 15 minutes.
3. Should be done standing up to help everyone participate.

During this time, the team members address three questions. The rules may change or be bent (see eg. Rising and Janoff, 2000; Yip, 2006), but the three

questions tend to be cited essentially intact (Rising and Janoff, 2000; Yip, 2006; Schwaber and Sutherland, 2011):

1. What did I do yesterday?
2. What will I do today?
3. Are there impediments to progress?

As the reader might agree, dailies are very simple to describe. Indeed, that is part of their allure as part of the agile toolkit, second in uptake only to iterations (West et al., 2010). We have thus described the essential constituents of a daily scrum, daily stand-up, daily meeting or henceforth just *daily*: discussing progress with everyone every day to focus and empower the team. We have also provided some clarification on how the daily relates to the texture of agile software development activities as a whole.

Due to its popularity, Scrum-inspired agile is reality to many in the industry. However, being a methodology, Scrum may leave you without direction should you wish to adapt its methods to suit your own context. In the next section we'll take a dive to the industrial heritage of agile and see if, instead of ascribed, empirically discovered methods, we can find something more fundamental.

6 The lean perspective

In section 5 we described how agile took off after the prominent failure of plan-driven methods to help with software development, and how Scrum and dailies relate to the picture in terms of everyday software development activities. In this section we're going to cover the field of *lean software development*. First, we'll see how lean has a shared heritage with agile. Secondly, we'll find out about how lean principles are applicable to software development, which will finally lead us to kanban systems as a method in software development.

6.1 Agile's lean heritage

The seminal work on lean software development is Lean Software Development by Mary and Tom Poppendieck (2003). We'll be using their description of lean to set the stage and see how the discourse is positioned in relation to agile.

In the book's foreword, Jim Highsmith and Ken Schwaber – both prominent signatories to the agile manifesto only some years earlier (Beck et al., 2001) – both individually rejoice at the possibility of applying lean industrial methods to software development (Poppendieck and Poppendieck, 2003, p. xiii-xvi). Highsmith takes note that while agile had an heritage in lean industrial methods, they were simply unapplicable before this work. Schwaber explains that agile processes were constructed "based on experience, trial-and-error, knowledge of what didn't work, and best practices", and that these new tools would allow agile practitioners "to understand why and how the most common agile processes work, or to modify them based on a deep understanding of them, or to construct their own agile process". (Poppendieck and Poppendieck, 2003, p. xiii-xvi)

What Schwaber is saying is that while he was able to put together a set of empirically discovered and validated methods, he was at a dead end when it came to finding a satisfying way of describing *why* they worked and what the constraints on their effectiveness would be. At the lack of the latter, adherents to Scrum would only have at hand a description of what to do, they'd be left fumbling if they discovered the results not to be to their liking. Indeed, you could be left with the idea that any deviations were harmful and your failure to find process nirvana was only due to failing at being strict enough. On the other hand, lean was pitched as reaching further towards the fundamentals of agile ideas, and allowing the discovery of new methods which would nevertheless serve to achieve the goals set by agile. Where Scrum had been a playbook detailing what was essentially black magic and the potential

to devolve into cargo cult[4] software development, lean could serve as a process construction kit.

6.2 Lean and software development

Poppendieck and Poppendieck (2003) observe that software development might have a lot to learn from new product development in general. They leverage principles of what is called *lean development* according to an approach used by automotive manufacturers Toyota and Honda starting from the 1980s[5], which they describe in the following fashion:

> Don't make irreversible decisions in the first place; delay design decisions as long as possible, and when they are made, make them with the best available information to make them correctly.

To these principles and other related qualities Poppendieck and Poppendieck (2003) attribute a significant competetive advantage exhibited by the Japanese manufacturers over U.S. ones. Advantage of this kind would ostensibly be desirable to any software company as well. What's good about this approach is that it doesn't come as a bundle of methods to apply, but as principles "understood and proven by managers in many disciplines outside of software development". Principles differ from methods by being universal guiding ideas and insights, which are however not easy to apply to particular environments.

The *seven lean principles* forming the basis of lean thinking are defined by

[4]Cargo Cult Science being famous from Richard Feynman's commencement address at Caltech 1974. He relates a story of an indigenous people who imitate the operation of a runway in hopes of making cargo planes land, yet none do. They're missing something essential. (Feynman, 1998)

[5]A more detailed story on the birth and popularization of lean manufacturing can be found in *e.g.* Ohno (1988).

Poppendieck and Poppendieck (2003) as follows.

1. **Eliminate waste.** Anything in the process not contributing to value as perceived by the customer is waste.
2. **Amplify learning.** Software development is a process of learning what works, not an exercise of reducing variation; trying out different approaches and seeing what works must be enabled.
3. **Decide as late as possible.** Deferring decisions is a requirement for effective strategy in complex and evolving environments.
4. **Deliver as fast as possible.** Fast delivery is critical from both customer value and feedback cycle perspectives.
5. **Empower the team.** The people who actually do the work are the ones who understand how to achieve excellence in execution.
6. **Build integrity in.** Software must be sound conceptually, be able to maintain its usefulness over time, and altogether be fit for purpose.
7. **See the whole.** Measuring the specialized contribution of individuals or organizations leads to suboptimization, which is antithetical to integrity.

Of these principles, the first is declared most important and the rest can be seen as following from them. To complete the picture and allow the reader to apply these principles to forming agile methods applicable to their context, Poppendieck and Poppendieck (2003) present 22 different tools. Some of these tools, like the use of iterations, refactoring and testing, will be readily familiar to most agile practitioners (West et al., 2010). We will be taking a further look at something less familiar – what on this list of tools is introduced as *pull systems*, otherwise known by the Japanese name kanban.

6.3 Kanban as a way for continuous improvement

Poppendieck and Poppendieck (2003) motivate pull systems as a way to address principle 4, *deliver as fast as possible*. The argumentation is profoundly simple: people showing up for work need a way to decide what to do. The alternatives are for you to order them to do something, or give them a way to discover that for themselves. In a fast-moving and complex environment, they argue, only the second option for work coordination is viable. This is attributed to the fact that in a chain of connected events, any variation is amplified, making any predefined scheduling invalid in short order. A pull system, therefore, is devised to make work items visible and enable self-direction such that team members may make the most productive use of their time. The name *kanban* for such an approach derives from Japanese for "card" or "placard" and refers to the tokens which stand in for work items.

In software development, Poppendieck and Poppendieck (2003) suggest, kanban may take the form of a board on which work item cards flow through a procession of steps defined by the team. This conceptualization of kanban could be argued to have little more to it than simply allowing developers themselves pick items from a backlog of work items (described in section 5.2). A literature review by Ahmad et al. (2013) points at the current conceptualization of kanban in the context of software development having been born in 2004. The following may be considered the kanban principles (Anderson, 2010), found to be congruent with the lean ones (Ahmad et al., 2013):

1. Visualize the workflow.
2. Limit work in progress.
3. Measure and manage flow.
4. Make process policies explicit.
5. Improve collaboratively.

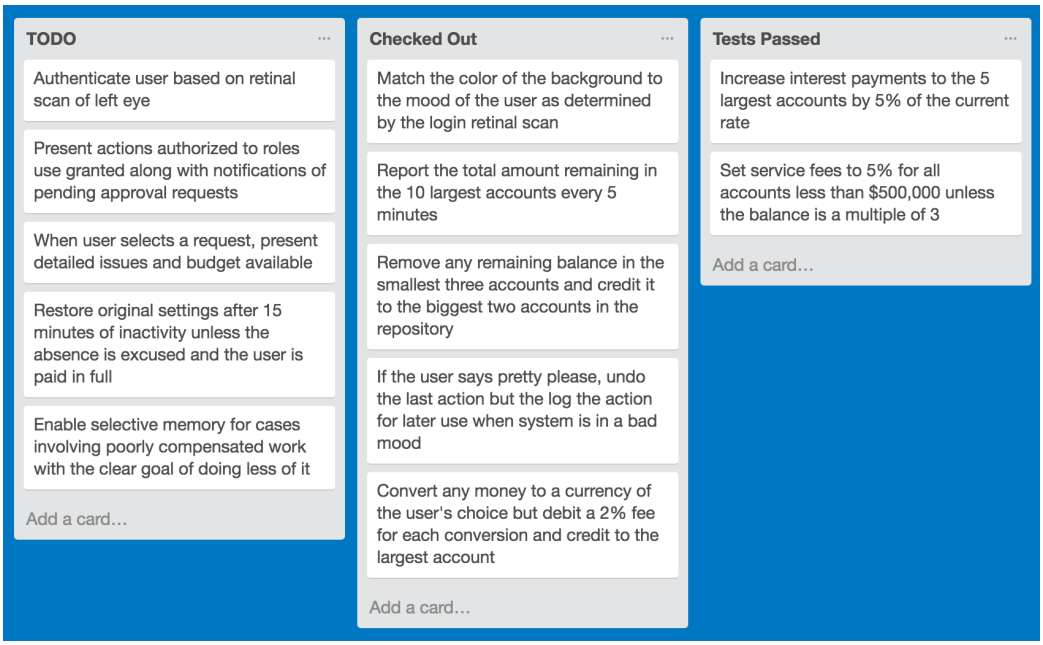

Figure 1: An example of a software kanban system, where tasks flow from left to right. Adapted from Poppendieck and Poppendieck (2003). The original figure has been reimplemented using the web-based Trello software (www.trello.com).

The clearest benefits Ahmad et al. (2013) can find attributed to use of kanban in the literature are better understanding of whole process, improved software quality and improved customer satisfaction. Challenges, on the other hand, include the fact that kanban is not a standalone measure but requires supporting methods (*e.g.* agile ones) and that it necessitates a difficult change in organisational culture. The potential for better understanding of the whole software development process is especially interesting, as Ikonen et al. (2010) suggest that kanban supports the ability to *see waste in the process*.

What happens when we take lean principles and find tools to implement them in practice can be described in more general terms. The drive for a collaborative and continuous elimination of waste may be taken as the defining characteristic of a process of *continuous improvement* (CI). That definition is posed by Bhuiyan and Baghel (2005), who in their literature review place lean manufacturing in a whole class of methodologies sharing the characteristic (six sigma, balanced scorecard, and lean six sigma being the others). It would take little imagination to claim that when lean methods are applied in software development, what they serve to enable is likewise continuous improvement.

Boer and Gertsen (2003) go further to define that when the CI from manufacturing context is overlaid with learning and innovation, it instead becomes *continuous innovation*. We see how, given the view by Poppendieck and Poppendieck (2003) that software development is inherently a learning activity, it could be argued that lean software development is by necessity an exercise in continuous innovation. On the other hand, the distinction can be seen as meaningless in the context of software development. In favor of staying within the lean discourse we will opt for the term continuous improvement to characterise this process.

We have described how lean software development is formed by a collection of principles, the most fundamental of which is waste reduction. The principles taken together aim at the collaborative activity of continuous improvement.

Practices derived from these principles may thus be taken to be, in effect, activities in a process of continuous improvement. Due to the nature of software development activities, this process will inherently involve learning and finding new ways of doing. Kanban is a method the basic use of which is work visualization and self-direction, but which can also be used as a way for seeing waste and through this be an instrument of continuous improvement.

7 Improvement in practice

In what we have described so far, the theme of improvement has been most central. Practitioners of Scrum wished to discover a way of doing things better than plan-driven methods allowed for, and lean principles direct a practitioner not only towards one solution but a way of doing things such that change is not merely tolerated but enabled and supported. In this vein we could go deeper and attempt to find out what it is that enables a practice to support continuous improvement. In the following we will draw from the theoretical toolbox of the practice-based approach in an attempt to find the necessary means of description for such a process and the means by which continuous improvement happens.

7.1 Improvement as a social process of innovation

Innovation can be taken as ideas, practices or artefacts perceived to be new to whomever is adopting them (Zaltman et al., 1973; Rogers and Shoemaker, 1971). Crucially, innovation may also be applied to the process through which new ideas are generated. Even though the word innovation is often attached to concrete objects, the key feature is simply novelty (Slappendel, 1996).

The improvement of ways of working requires taking novel approaches distinct

from how things are done in the present state. Should we take software development to be a collaborative activity where innovations are required to deliver novel solutions, we may also consider the process of delivering or aiding the delivery thereof as something that may be innovated on. Innovation in social interactions has been studied in *innovative knowledge communities*. Hakkarainen et al. (2004) outline three different perspectives on them:

1. **SECI model** (Nonaka and Takeuchi, 1995). Knowledge is something that can be explicated and objectified. Innovation happens by transforming tacit knowledge to explicit.
2. **Activity theory** (Engeström et al., 1999). Knowledge is embedded within practices. Overcoming tensions inherent in the activity is a source of innovation.
3. **Conceptual artefacts** (Bereiter, 2002). Knowledge is expanded by manipulation of shared conceptual artefacts. The ability to extend and create novel artefacts is the source of innovation.

Out of these three perspectives, it is Engestrom's that is best suited for describing improvement in ways of working. If knowledge is embedded in practices (Engeström et al., 1999), then innovation must be the development of those practices. Engestrom is specifically looking at communities where innovation is the development of practices with a shared object of activity, which meshes well with the idea of an agile team working with a kanban system. Also directly relating to the goal of the study, Engeström (2000) claims activity theory may be used as a tool for analyzing and redesigning work. It is stated to be especially applicable when such learning occurs which has not been given as the objective from the outside, such as by management, but instead stems from contradictory demands experienced within the community (Engeström, 2001; Engeström and Sannino, 2010).

The most prominent conjunction of activity theory and software development so far has been in the field of human-computer interaction research, for exam-

ple Kuutti (1996) and Kaptelinin (1996). These works mainly take interest on what happens in the interface between human and computer, in the context of an activity system, instead of on what people do together. Barthelmess and Anderson (2002) and Adler (2005), however, take an activity theoretical view on the process of software development and show that the perspective is applicable for describing some of the industry's most problematic features. In a study on software development environments, Barthelmess and Anderson (2002) motivate the use of activity theory by characterising the process of software development as collaborative effort around complex, intangible artefacts. In doing so, they claim to make a break from typically "production-oriented" descriptions of software engineering. Adler (2005) characterise software development as non-routine knowledge-work, the nature of which is affected by the model used to describe it. Activity theoretical concepts are used to describe how a specific modelling may add to the challenge of responding to contradictory demands (Adler, 2005, p. 421).

The activity theoretical approach appears to be a good fit for describing CI based on what we found out in section 6.3. We will be looking at continuous improvement as taking place within the context of an interactive, social process of innovation as occurring on the level of a group of people having the same object of activity. In the empirical study we will be taking this view and applying a strategy inspired by Nicolini (2009), where we *zoom in and out* of a practice and inspect it on different levels - from patterns of interactions between individuals to the wider texture of related practices. In the next section we will familiarize ourselves with activity theory and how it relates to our focus of investigation, innovation.

7.2 Activity theoretical perspective on innovation

Activity theory can be seen as a general paradigm of studying transformations and social processes, where different activities are distinguished by their objects (Kerosuo et al., 2010). The object of an activity is the reason the activity exists; without the object of work the activity would cease (Engeström, 2000). An activity system may be seen as a sensemaking device where the "raw" object is, as the outcome of a process, transformed into another, collectively understood object (Engeström, 2001).

Engeström (2001) outlines the principles of activity theory as such:

1. Activity system as the prime unit of analysis
2. Multi-voicedness
3. Historicity
4. Contradictions as sources of change
5. Possibility of expansive transformations

Activity theory concerns the analysis of activity systems which are oriented towards an object of activity, mediated by artefacts and collaborative in nature. Multi-voicedness stems from the fact that in a collaborative effort there are always multiple viewpoints; division of labor serves to create different viewpoints for collaborators with different personal histories. The system itself is historical, too, with rules, conventions and artefacts to attest to that. Stemming from historicity and interactions with an environment, activity systems may be seen as residing inherently in a perpetual state of contradiction, structural tensions between aspects of the system. Finally, it's recognized that these contradictions may be aggravated such that qualitative, expansive transformations in the activity system takes place.

An activity system has a few interesting features in terms of change, learning and innovation. Historicity, the fact that before this state of the activity

system there was another state that looked a bit different, implies that the system can get out of date with regard to objects and other systems it interfaces with. It is these expansive transformations which constitute innovation in the terms we lined out in section 7.1.

Innovation tends to occur when there is pressure to change. Within practices, this pressure may be attributed to a "functional failure" of the practice (Miettinen, 2006). When the world changes, the way the practice used to work suddenly might not. Engeström (1987, 2001) describe these situations as developmental contradictions in the practice that generate disturbances. Disturbances are episodes of deviation from standard script (Engeström et al., 1996; Norros, 1996) or, simply, the breaking of a habit, and serve as indication that the system exhibits change potential.

Expressing contradictions alone is not sufficient for change. Contradictions surfacing as disturbances need to be grasped and taken as a motive (Engeström and Sannino, 2010). Transformations in the practice may result from deliberate collaborative change effort within cycles of expansive learning (Engeström, 2001). In activity systems, learning actions in such cycles are the means by which contradictions may be recognized and addressed (Engeström, 2000; Engeström and Sannino, 2010). Learning actions in the "ideal" cycle of expansive learning have been linked to four tiers of contradictions as follows (Engeström, 2001; Engeström and Sannino, 2010):

1. **Questioning.** Some parts of accepted practice are questioned, criticized, or rejected. (Stems from a *primary contradiction*.)
2. **Analyzing.** Transformation of the situation to find out causes or explanations. (*Secondary contradictions* are presented.)
3. **Modeling.** A simplified, explicit model of the explanatory relationship may be offered.
4. **Examining the model.** The model's applicability is tested.
5. **Implementing the model.** Practical applications, enrichment and

conceptual extension bring the model to life. (*Tertiary contradictions* manifest as resistance from other parts of practice.)

6. **Reflection.** The reality of the implemented model's viability is observed.
7. **Consolidation.** The new practice becomes stable and is aligned with neighbouring practices. (*Quaternary contradictions* occur due to realignment.)

From this view, change in a practice amounts to the outcome of a pressure exerted by the existence of a primary contradiction. This pressure is resisted by the other contradictions. There may not be an evident transformation of the situation such that the contradiction would be resolved acceptably. Attempting to implement such a transformation might require adjustment to other parts of practice, so the group needs to undertake effort not initially visioned. Finally, even if this group of practitioners finds something that works for them, the surrounding network of activity systems will need to deal with the effects too.

The different levels of contradiction present a way of mapping arising disturbances and contradictions to a typology where they can be distinguished as being connected to different stages in the expansive learning process. In the empirical analysis section we will be leveraging the following characterisation of the manifestation of contradictions in practice, adapted from Yamagata-Lynch and Haudenschild (2009):

Primary Practitioners encounter contradictory demands attached to an element of practice.

Secondary Practitioners encounter a new element of activity, and the process of assimilating the new element into the practice generates conflict.

Tertiary Practitioners encounter a newly advanced method for achieving the object of a practice which is in conflict with

the existing state of practice.

Quaternary Practitioners encounter changes to the practice that are in conflict with neighboring practices.

Expansive learning is thus to be understood as a process of construction and resolution of contradictions that follow from each other. These actions don't, however, need to follow each other in temporal order, and they don't all need to be present for learning to occur. Also noteworthy is the fact that it is not *individuals* who change, but instead the object of collective activity and thus the components of the activity system linked to it. (Engeström and Sannino, 2010)

If we are to understand innovation, we must start with seeing the shared object of activity as its basis (Miettinen, 2006). Furthermore, in order to understand the an expansive learning process in action, we need to gain an understanding of contradictions and their transformations in the activity (Kerosuo et al., 2010). This will be the undertaking awaiting us in the empirical research chapter.

8 Theoretical synthesis

This section details answers to theoretical research questions posed in section 2 and sets further research questions as the basis for research in the empirical study.

8.1 Answers to theoretical research questions

> **TRQ1:** How does a daily in agile software development support the development team?

Agile software development refers to a class of software development methodologies, which may be seen as belonging to a particular culture or perhaps even ideological movement. Agile may be characterised by incremental software delivery, cooperativity between developers and clients, straightforwardness and adaptability, and adaptivity to changing circumstances. The daily belongs among the most popular of agile methods in practice. It was coined by early practitioners of the Scrum methodology, also belonging to the strongest influencers in the industry.

The daily is a short, all-hands meeting held every day. Dailies occur in the context of sprints, development cycles which last for a preset time on the order of weeks. A sprint has a predefined list of work items assigned to it, representing the most important requirements to be satisfied. In the daily, participants discuss the progress of work items and whether help is needed to resolve impediments to progress. An agile daily thus supports the development team by providing a daily ritual which empowers the team members to aid each other and helps them focus on the essential in the context of a sprint with defined goals.

> **TRQ2:** How does the use of a kanban system support continuous improvement in the context of lean software development?

Lean software development is an application of lean development principles, originally used in the automotive industry, to the field of software development. The first principle, which may be taken to be the goal of lean, is the elimination of waste. Waste is everything not contributing to value seen by the customer. Software development, on the other hand, is necessarily an effort of learning or discover better ways to build things. Finally, continuous improvement is the collaborative and unending effort to eliminate waste.

In rapidly changing and complex environments, work coordination is a difficult problem. Also, to leverage input from feedback cycles for learning, delivery should be as fast as possible. For fast value delivery, it is essential that workers be able to interpret by themselves what work there is to be done. A kanban system in software development is used to visualize the workflow and manage the flow of work items through it. Such a system facilitates understanding of the process and being able to see waste in the process as it exists.

The use of a kanban system in lean software development supports continuous improvement in two ways. Firstly, it directly supports faster value delivery, which contributes to continuous improvement by enabling learning. Secondly, it supports seeing waste in the workflow, which potentially allows for addressing the generation of that waste.

> **TRQ3:** How can continuous improvement be observed in practice?

Improvement in general may be seen as a social process of innovation, which can apply to practices. Innovation in social interactions has been studied in innovative knowledge communities. Activity theory is a practice-based approach with which we can look into innovation.

Activity theory describes practices as historical, object-oriented activity systems. Collaboration in the system is artefact-mediated and bounded by rules and division of labor in a community of practice. Activity systems are in a perpetual state of contradiction, or structural tension between parts of practice. Innovation in activity systems is understood as expansive transformation of the object of activity, whereby these tensions are resolved.

Contradictions are the source of innovation. They arise in practice as errors, disturbances or discontent, which indicate that the system exhibits change potential. Primary contradictions are a questioning of a part of accepted practice, exerting pressure on the current state of practice by pointing out a failing. By offering explanations and analysis, secondary contradictions opposing the change pressure can present themselves. If a suitable transformation of the situation is found, tertiary contradictions occur when the new part of practice is fitted with its surrounding system. Finally, a model of action found viable will have to deal with other interfacing practices, and quaternary contradictions can be observed due to the realignment required. Expansive learning is thus a progression of constructing and resolving contradictions, but these do not need to occur in any specific temporal order.

Activity theory may be used to observe continuous improvement in practice by seeing the practice as an activity system, paying attention to developmental contradictions that occur in the practice, and investigating any resulting transformations in the object of activity.

8.2 Empirical research questions

The theoretical synthesis detailed the role of a daily in the Scrum methodology, the means by which use of kanban can contribute to continuous improvement, and how activity theory may be used to observe continuous improvement in practice. Based on the synthesis, the following empirical research questions (ERQs) are posed to the empirical research data.

> **ERQ1**: How is the observed daily positioned with regard to descriptions of Scrum daily and kanban systems in the theory?
>
> **ERQ2**: How do contradictions and their transformations manifest in the observed dailies?

With ERQ1 we will establish the observed daily practice, which will allow us to position the practice in relation to the theory of Scrum dailies and kanban systems. We will elucidate the aspects by which the observed practice exemplifies what the theory says should occur, and the aspects by which the practice differs from theory. The practice will be described in terms of being an activity system, especially by the definition of its object of activity. Finally, we will be left with hints on what *is not* static in this practice and what we should thus pay attention to when looking for clues of continuous improvement.

The establishment of a normal state of affairs is conducted in preparation for ERQ2. In the second part of the empirical analysis we will observe how the regular flow of the practice is *disturbed*, taken off-script. Particularly, we will be able to show that disturbances bring forth contradictions of different kinds, as implied by the theory of developmental contradictions. We will show that there exist *transformations* in the object of activity, and will be able to describe these transformations as potentially happening as the result of a number of interrelated episodes where contradictions of different kinds

have been brought forth.

The undertaking is first to to describe the daily as a practice within software development, serving to improve on the team's understanding of its workflows and being linked to the exercise of creating software. We will further show that the daily is a platform that supports surfacing contradictions considering the team's workflow. Thus we will be able to conclude that there is use for the daily as a mechanism that supports continuous improvement of a software development team – or put more simply, *improving on how the work works*.

Chapter III

Empirical study

This chapter contains description of the empirical case study conducted for this thesis. Section 9 describes the research context, setting and participants. Data collection and analysis is covered in section 10. Finally, empirical findings based on the theoretical perspective from chapter II are delineated in section 11.

9 Empirical study description

This thesis uses empirical data collected ethnographically from daily activities in a software development team. Section 9.1 describes the research context and the researcher's role in it. The environment of the daily as well as roles taken and tools used are detailed in section 9.2.

9.1 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 a virtual kanban system every morning was taken as an opportunity to learn about the daily method as practiced within the context of a lean workflow.

The researcher took part in the development team's dailies having two roles. The role of an ordinary team member necessitated normal interaction with the team within the context of the daily. The role of a researcher involved recording the proceedings and logging observations directly after the event.

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* 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.

9.2 Daily setting, roles, tools and activities

The development team daily occurred every day after a short whole-company standup meeting scheduled to start at 10 in the morning. The context of the observed dailies was 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*.

The driver and the secretary are the only explicit roles defined in the context of the observed daily. Both may be seen as facilitators for the daily, in that the driver presents a de facto agenda and the secretary ensures that items on the agenda, or any emerging ones, reach an outcome satisfactory for the team.

The driver connects their laptop to the TV screen, enabling the team to view their virtual kanban system and work items. Throughout the daily the driver will work through the kanban system and work items according to how the team's discussion proceeds. Usually, the driver will walk the team through boards and their work items in the order that they appear in. The team might make callouts and request that the driver deviate from this, and if the team's discussion diverges to a work item not currently shown the driver will attempt to accommodate.

The secretary puts into effect 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. Functioning in the role of the secretary necessitates that either the team make their decisions explicit or that the secretary intervene to clarify the outcomes from any discussion.

The most important tool made use of was Asana[1], a web-based task management application in which the team had encoded their workflow. It views work as *projects*, which consist of *tasks*, 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 team modelled their workflow with the use of projects and labels and moved tasks

[1] www.asana.com

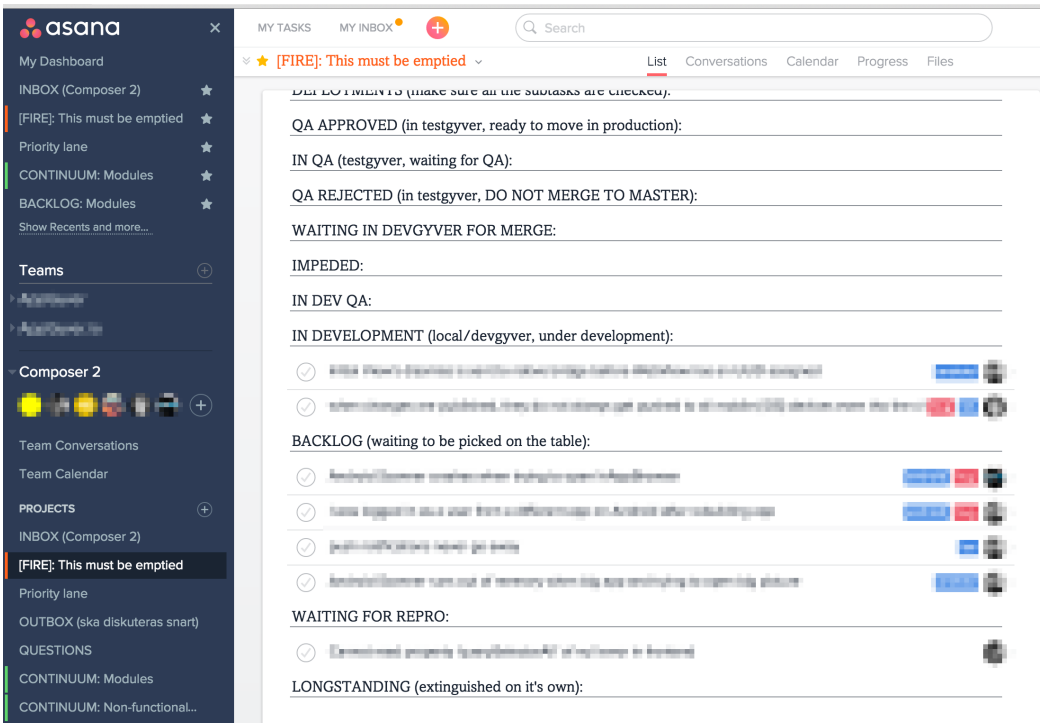

Figure 2: Asana. On the bottom left is a list of the team's kanban boards. On the right, a single board with its labels and tasks can be seen.

in between them. According to the example on use of kanban from section 6.3, in this study Asana projects were taken to be virtual kanban boards, segments were taken to be phases in the process and tasks were understood as kanban tokens for work items.

In addition, meeting minutes were published through Flowdock[2], a company-internal instant messaging and group chat application. The application structures communication in terms of *flows*, persistent message logs with a certain topic. One of these flows was dedicated for meeting minutes from dailies and used by the secretary during the daily.

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

[2]www.flowdock.com

10 Data collection and analysis

The following sections present the empirical data used for this thesis, basing on the empirical study description from section 9. Section 10.1 describes how the research was undertaken and the data analysed. An overview of the resulting body of empirical data is provided in section 10.2.

10.1 Research and data analysis 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 researcher took part in all of these events and took the same role of secretary (as described in section 9.2) 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 primary data for this research, as defined in section 3. After dailies, notes were made to accompany the recordings as basis for further analysis. 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 9.2. These notes and screenshots constitute secondary data which were used to validate and support

understanding of primary data.

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.

Capturing the kanban boards' structure as screenshots yielded material of two different kinds. As described in section 9.2, the team's kanban boards were accessed through a web-based UI. In this UI, kanban boards were arranged as a list of text entries. The contents of a kanban board being viewed, tasks and task states, likewise appeared as a list. Lists of kanban boards were extracted and are presented as indexes in appendix B2. Individual kanban boards' structure, meaning the list of states used for tasks, were extracted in a similar fashion and presented fully in appendix B3.

The content log and board structures afforded the completion of a *segment analysis*, a coding process arriving at a structural outline of dailies which provided an understanding of broad patterns of activity as accorded with section 3. When attention to narrow episodic fragments was called for, they were transcribed individually in more accurate detail. Such transcriptions will be presented in this study following this convention:

Timestamp in minutes:seconds "Speech in between quote marks"
(with `attributions` in parentheses)
(descriptions, interpretations and characterisations of situation or action in parentheses)
"Omissions of fragments [..] and injections [of] missing parts of speech for understandability marked by square brackets"

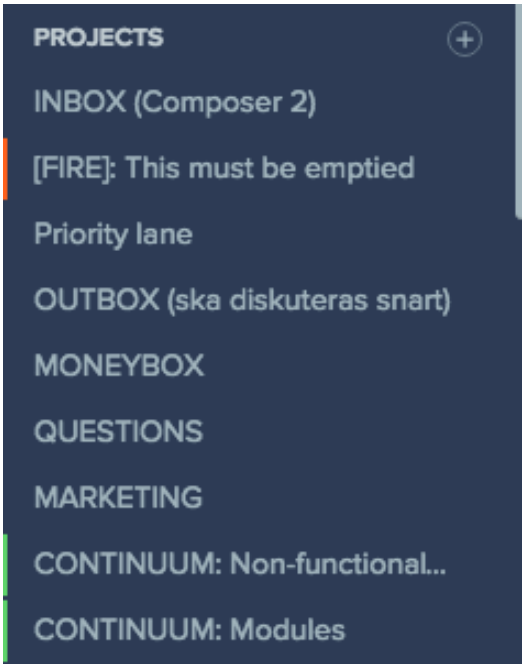

Figure 3: A captured list of kanban boards on Asana.

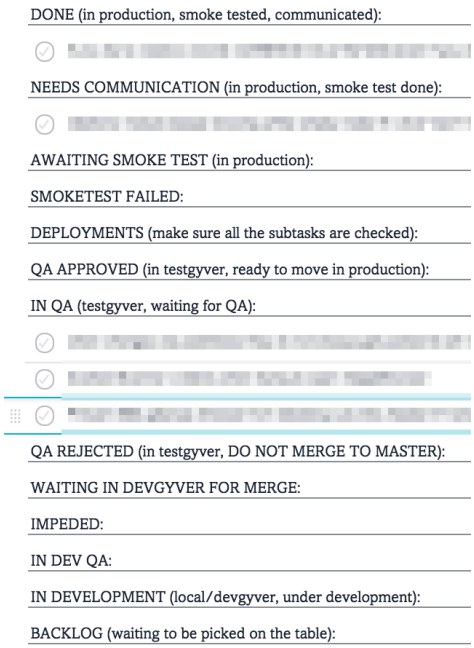

Figure 4: Excerpt of a captured list of kanban board states on Asana. The blurred parts are individual tasks.

Timestamp for next set of actions (turns of speech taken in continuous fashion can be transcribed under a single timestamp, but otherwise individual timestamps are presented)

For the sake of both succinctness and preservation of anonymity, when attributions are provided 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. As a further anonymity preservation measure, indexes used are specific to the transcription fragment instead of being persistent over fragments.

The audio recordings were further reviewed with the aid of the content log to discover episodes of particular theoretical interest. Firstly, fragments of the content log were curated on new documents, one per each daily, as potentially pointing to interesting details. Then, the audio referenced in the content log fragment was reviewed and its relative importance within the body of material reconsidered. Fragments found theoretically relevant were retained in the curated documents. Some fragments were seen to benefit from more in-depth understanding and were transcribed. The episode catalogue thus formed constituted material for analysis of disturbance episodes.

The final part of the empirical analysis was enabled by a joining of the theoretical understanding of the importance and nature of transformational processes, expounded in 7.2, and the created episode catalogue. The episodes were further curated such that some of them could be linked to each other thematically by the researcher. Of the altogether 10 possible storylines identified, one was sampled for complete transcription and presentation in 11.4 on the basis of scope and importance to the team's daily activities.

10.2 Overview of collected data

This section presents an outline on the data and its salient features. The recorded dailies are first described in terms of duration, attendance and language used. Then the picture is detailed by describing segments occurring within the dailies.

10.2.1 Daily duration, attendance and language

The 20 recorded dailies 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. Table 2 is introduced as a basis for reference.

The dailies gathered an attendance of three to ten participants. 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`).

Due to 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`). Quotes originally in Finnish are presented as translated to English.

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	≥ 8
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	≥ 5
09	18.12.2015	19:03	17:20	Finnish	≥ 5
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	≥ 8
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 2: 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. Attendance numbers marked with $\geq$ are lower bounds deduced from recordings in scenarios where the number was not reported in daily memos.

10.2.2 Identified segments and terminology

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

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

Segment	Total time spent	Average time spent
INBOX	04:04:35	00:11:39
FIRES	02:18:41	00:06:36
INTRO	01:46:30	00:05:55
OUTBOX	01:21:28	00:04:04
NON-FUNCTIONAL CONTINUUM	00:24:44	00:01:54
PRIORITY LANE	00:24:17	00:01:26
QUESTIONS	00:22:21	00:01:15
BOARD OVERVIEW	00:20:38	00:06:53
CYCLES	00:15:02	00:07:31
NON-CORE MODULE CONTINUUM	00:13:25	00:01:41
MONEYBOX	00:00:49	00:00:25
MODULES CONTINUUM	00:00:25	00:00:25
MARKETING BOX	00:00:13	00:00:13
WATBOX	00:00:13	00:00:13
Totals	11:33:21	00:04:45

Unless otherwise specified, a segment is named by reference to a concrete instance of a board. In these segments, the team's attention is focused on a specific board. Prominent examples include `INBOX` and `FIRES`. These are considered participant constructs and attempt is made to present them faithfully as such, retaining original names even if they are highly idiomatic (such as `WATBOX`). Other segments are synthetic: they were identified and named by the researcher, or grouped together from segments ranging over multiple boards.

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. 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. `INTRO` and `BOARD OVERVIEW` are both synthetic, non-participant constructs.

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. *Cycle* is the team's name for a single development cadence. The opposite of a cycle is a *continuum*, evident in boards such as `NON-CORE MODULE CONTINUUM` and `NON-FUNCTIONAL CONTINUUM`. Work completed on such boards can generally be shipped to production immediately after completion.

Cycles are used for sets of features that are to be completed in synchrony with each other and that cannot be shipped incrementally. Cycles 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. The board for a specific cycle will vanish from sight by being archived after work in it has been completed. All of these segments were grouped together as `CYCLES`. Discussion generally involves *what is being worked on*, *by whom* and *when it will be completed*.

11 Empirical findings

This section presents empirical findings based on which we will answer the empirical research questions from section 8.2. To answer **ERQ1**, we describe the structure of a daily and link change in that daily structure to corresponding change in kanban boards in sections 11.1 and 11.2. **ERQ2** is answered by investigating activity episodes where contradictions occur in section 11.3, and linking episodes to a journey of expansive learning in section 11.4.

11.1 Typical daily segments

An overview of the structure of a daily is afforded to us by the segment analysis described in 10.1. No two dailies followed exactly the same sequence of segments, as the reader can observe from appendix A1. However, clear tendencies could be seen. In this section we first describe a prototypical example of a daily and proceed to paint a picture of how the daily came to be structured the way we 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 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	00:00:41
NON-FUNCTIONAL CONTINUUM	00:01:19
BOARD OVERVIEW	00:02:13
Total duration	00:28:18

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

Introduction, 00:40. 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 reassurances 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.

Inbox, 11:25. The INBOX segment finally starts. 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.

Fires, 13:10. 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 completion 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.

Priority lane, 17:15. The driver opens up the next board. The flow of discussion regarding `PRIORITY LANE` is similar to `FIRES`, 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`, adding 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.

Outbox, 20:45. The driver opens up *"OUTBOX (ska diskuteras snart)"* and commences the `OUTBOX` segment 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 getting an affirmative response from the secretary. This concludes the OUTBOX segment.

Questions, 24:05. 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.

Non-functional continuum, 24:46. 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.

Board overview, 26:05. 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.

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

Nobody objects; the daily is concluded.

11.2 Change in the daily structure

The previous section answered the question of how the daily happened. To support an argument about a relation between the daily and its sociomaterial context, in this section we attempt to reach for an understanding on how the daily could have come to take this form.

Observing table 5 where the segments undergone in the daily and the kanban boards have been overlaid, we find that the boards are in the same order as the team uses when proceeding through the daily. This gives rise to the idea that the structure of a daily and the structure of the boards are interrelated. Indeed, this is not an uncommon occurrence: `INBOX` – `FIRES` - `PRIORITY LANE` - `OUTBOX` - `QUESTIONS` is a typical sequence through the observation period.

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 5: List of daily segments and the kanban board list for daily `03` overlaid.

Table 6 details occurrences of daily segments. The segments `INBOX`, `FIRES`, `PRIORITY LANE`, `OUTBOX` and `QUESTIONS` occur from 17 to 21 times. We could consider these top segments to form the stable skeleton of a daily. Most other segments only get a handful of visits, with the runner-ups being `NON-FUNCTIONAL CONTINUUM` at 13 and `NON-CORE MODULE CONTINUUM` at 8 occurrences respectively.

Table 6: Total occurrences of segments through observation period.

Segment	Occurrences
FIRES	21
INBOX	21
OUTBOX	20
INTRO	18
QUESTIONS	18
PRIORITY LANE	17
NON-FUNCTIONAL CONTINUUM	13
NON-CORE MODULE CONTINUUM	8
BOARD OVERVIEW	3
CYCLES	2
MONEYBOX	2
MARKETING BOX	1
MODULES CONTINUUM	1
WATBOX	1
Total	146

It might be reasonable to infer a causality here. However, it's not evident if the team's daily structure is indeed encoded in the kanban boards' layout, or if the boards are merely arranged in imitation of the daily. To answer this question, consider what happens when the boards are disturbed. Table 6 gives a hint about where to look for such a disturbance: segments `MONEYBOX`, `MARKETING BOX` and `WATBOX` appear for the first time at the end of the observation period

but are nowhere to be seen by daily `20`. To corroborate a hypothesis about a relation between the daily and the boards, a first order explanation of a change having been made to the boards would suffice. Table 7 describes modifications made to the board layout for daily `19`.

Prior to daily `19`	Daily `19`
INBOX (Composer 2)	INBOX (Composer 2)
[FIRE]: This must be emptied	[FIRE]: This must be emptied
Priority lane	Priority lane
OUTBOX (ska diskuteras snart)	OUTBOX (ska diskuteras snart)
-	MONEYBOX
QUESTIONS	QUESTIONS
-	MARKETING
CONTINUUM: Non-functional	CONTINUUM: Non-functional
CONTINUUM: Modules	CONTINUUM: Modules

Table 7: Difference in board layout for daily `19`.

By daily `19`, boards associated with the segments `MONEYBOX` and `MARKETING BOX` appear in the layout, intermingled among the most commonly used boards. `WATBOX` seems to be an anomaly, because although it is exhibited in the daily segment after `MONEYBOX` the board itself is still out of the way and would not even be visible on the driver's screen. Apparently there was a commonality between the three, however, because in daily `20` the team has again groomed the boards for `MONEYBOX` and `MARKETING BOX` outside the group of most common boards and together with `WATBOX`.

One interpretation for this series of events is that there occurred a disturbance

which necessitated the provision of new places to put things for them to get done eg. at the appropriate timeliness. The boards are perused once or twice in dailies until the team figures that it's not getting any use out of doing this commonally, at which point the boards are moved out of the way of the ordinary daily flow.

From the looks of things, it does stand to reason that the structure of the board and the daily go hand in hand. If this is a credible statement, it should be a hint that the daily and its sociomaterial context are engaged in a recursive loop of unfolding driven by tensions introduced and resolved during the practice of a daily. It's too early to make that claim, however. Of course, we're only investigating the matter on the level of segments and boards and ignoring most of the story. In the next section we will take the chance to have a closer look at the dynamics and nature of this processual evolution more closely.

11.3 Episodes of contradiction

In section 11.1 we looked into how a daily happens and what kinds of activities a daily constitutes. Section 11.2 linked structural change in a daily to change in kanban boards, but the question of how that change happens was left open. Ground will be laid for answering **ERQ2** by investigating contradictions as the source of change. In section 7.2 we identified four different kinds of contradictions, and in this section, occurrences of the different contradictions are exemplified through individual episodes.

The first episode represents a primary contradiction with an immediate transformation.

Daily 04, `NON-FUNCTIONAL`

30:40 "If we are skipping modules all the time, then we need to

move non-functional on top of modules" (M)

30:46 "Makes sense" (D)

30:47 "Nice" (M)

A manager observes that the team's kanban boards are ordered such that the driver would be inclined to direct the team to the module continuum board at this point of the daily. However, going through that board is not seen to be of value for the team. The manager suggests that the board be moved out of the way, which the team accepts.

In the previous simple scenario a new modelling of the situation was apparent. Not all primary contradictions afford such easy resolution. In the following fragment, two developers have an exchange reflecting a more fundamental tension.

Daily 14, FIRES

14:05 "D1, this [fire] was here yesterday. Did you do anything about this?" (D2)
"No" (D1)
"Why not? Fires go always first." (D2)
"Yeah, but I was also working on this [feature]" (D1)
"But that's not a fire" (D2)
"But it was in a stage [where] I didn't want to cut the work" (D1)

The very existence of work items categorised as fires is contradictory to the team's goals, as they call for attention and effort to be redirected. However, switching tasks from a feature under development represents a big cost, and apparently in this scenario a developer judged the cost too high. The other developer has no alternative but to accept the existence of these contradictory demands. In this case, there is no transformation to be suggested.

The following fragment exemplifies a secondary contradiction. The team has encountered a swath of new work items and has found that most of them are unaddressable collaboratively by the team.

Daily 15, `INBOX`

22:18 "Would it make sense that we had two inboxes?" (`M`)
"Yes" (`D1`)
"Like, one for the modules because here, half of the issues are module [...]" (`M`)
22:30 "Then everyone has to know whether they are module issues or not" (`D2`)

The team's way of working through the inbox items is unsatisfactory at the face of these specific items. A manager suggests an immediate resolution such that the inbox be split in two, which would allow the team not to spend time on attending to work items that they "know" they cannot address. A developer notes that parties external to the team would need to readjust and learn how to perform this splitting, which would represent a quaternary contradiction, should the suggested transformation take place. However, the rest of the discussion reveals that this new influx of work items is desirable by at least part of the team.

24:19 "I think it's nice for everyone to see how broken modules are or are not" (`D`)
laughter (`M`)
24:28 "I put my stuff here [..] just so you can see 'ok this thing is broken' –" (`D`)
24:40 "Ok, let's continue. Can't optimize everything." (`M`)

The manager concedes that the situation does not have an ambiguous resolution. No transformation follows despite the earlier suggestion which was considered valid by part of the team.

The next fragment is familiar from daily `03`, presented earlier in section 11.1. A developer is explaining a failure discovered in production. Previous to this point in the discussion, the developer has taken measures to address a concern by a product owner and announced to the team that work items that are *fires* but related to *modules* shall be presented in the "module continuum" board. This new method conflicts with the way the team has previously modelled the timeliness requirements of fires as opposed to other tasks, and represents a tertiary contradiction.

Daily 03, `PRIORITY LANE`

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

18:03 "[..] I'm doing everything I can, I have the one other fire first. [..]" (`D`)

18:13 "[..] where is this kind of task?" (`M`)

18:16 "It's in the module continuum. [..]" (`D`)

18:23 "Okay... but wouldn't it make sense to promote them as a fires also? And not leave them in the module continuum." (`M`)

The developer has, while applying a new approach suggested by the PO and presented earlier, neglected to account for existing practices. The resolution suggested is to adhere to the existing convention of visualizing fires while also applying the new one.

The final disturbance presented here will feature a quaternary contradiction, one that represents a challenge in aligning the practices of this team and another. The team is in the process of shipping a bundle of features, which would require the cooperation from a team that is seen to be difficult to reach out to.

Daily 16, `OUTBOX`

14:10 "'[Other team] is almost but not quite [..] hard to reach'" (`D1` reading out item)

14:48 "It was [..] hard for me to go these through with [representative from other team] because we had to go really deep .. into .. *whose* responsibility it is to discuss with `M1` and everything. [..]" (`M2`)

15:08 "And are we getting our modules released and stuff like that?" (`D1`)

"No [..] we're getting [entirely unrelated things]" (`M2`)

15:16 "We're getting modules released by bypassing [the other team]" (`D2`)

"So they're still [..] hard to reach?" (`D1`)

"Yes. But we've actually agreed to a weekly meeting regarding modules with [the other team]" (`D2`)

15:35 "Until that has worked for twelve weeks in a row... we have a counter." (`M2`)

"So how about putting that to longstanding?" (`D1`)

The situation appears to be tentatively resolved by a scheduled coming-together of the two teams. Whether or not the realignment is successful is to be decided, however, and the team decides to leave a reminder in the longstanding section of the outbox for tracking the outcome.

In this section we have seen disturbances in the daily practice that could be interpreted as reflecting contradictions of all the tiers from primary to quaternary, and also seen how a transformation to resolve the contradiction may nor may not come up and likewise may or may not be accepted. Furthermore, some of the disturbances were clearly linked to how things are *done*, but others merely on how the doing is *modelled*.

The observations go to show that disturbances indeed do occur, and that they

may be linked to transformations. However, we might collect and catalogue all of the individual disturbances but still not be much better off in our capability to understand the usefulness of them to the team. As described in section 7.2, expansive transformations are supposed to happen over periods of time and as successive episodes where contradictions are brought up and resolved. In the next section we'll be looking for evidence of just such a process.

11.4 A transformational journey

In the introductory section the usefulness of a daily was set to question by the fact that it is not a software development activity in itself. To reiterate, no features are delivered, no bugs fixed and no customer served. Therefore, if a use for the daily is to be found, in that it somehow aids software development activities, the use must lie in the relation between the daily and other practices that *do* constitute software development.

From the episodes found in the empirical data, a set was chosen such that their orientation is *not* toward the daily and what happens within but impactful on the outside world. They were curated into journeys, episodes which appeared related through their contradictions and their transformations. In this section, we follow one of those journeys in depth. We're going to find out about what was discussed in 11.2, segments which prominently appeared and then vanished at the latter part of the observation period.

11.4.1 Trouble with work item categorization

The journey starts with the first time that the team can be observed to identify a problem with the way they handle inbox items. The standard procedure options apparent to the team are handling the new work item as either wishlist or fire. Here, they come across a work item that does not fit

their conception of either. The item appears to the team as simultaneously less urgent than a fire yet more important than a wishlist item.

Daily 09, `INBOX`

05:10 "Let's put this [new work item] on the 'required' list" (`D1`)
05:11 "Okay, so... wishlist?" (`D2`)
05:18 "This is a funny thing to be putting on a wishlist" (`D2`)
05:18 "But at least it's not a fire" (`D1`)

The team would not like to apply the standard procedures in place - either of them would be "a funny thing" to be doing. Taking note of this allows a latent primary contradiction to be surfaced. However, it is seen as merely a disturbance and no attempt at explanation or transformation follows: the resolution seems to be simply inaction, with the item left in the `INBOX` for later.

It takes a while for the issue to brew and mature. Five observed dailies (more than three weeks of calendar time) later we find the team discussing an outbox item which presents a similar conundrum. A work item appears which, paradoxically, will not need doing nor priorisation through wishlist until an additional external trigger.

Daily 14, `OUTBOX`

21:33 "I guess wishlist or something would be inappropriate?" (`M1`, presenting wishlist as a precluded option)
21:50 "Does [stakeholder] need the feature? ... Where is that feature?" (`D1`)
"There's a follow-up feature request that doesn't need to be done." (`D2`)
22:05 "Has it been through `INBOX`?" (`D1`)
"No. It's because that doesn't need to be done." (`D2`)

27:00 "It says 'pending project, will be done when customer project requires'" (D2)

27:15 "It's pending a project. There's no ... case." (M1, restating the intent after attention on the item is called for again)

"Can we throw this to [a release cycle]?" (D3, providing a further option)

"No, no, we won't be doing it" (D1 with what can be taken as a note of frustration)

This time, instead of being satisfied with their declaration of an item being unfit for the apparent categories, the team does attempt to model the situation such that a suitable place could be found. The team's action demonstrates the existence and attempted resolution of a secondary contradiction, which goes as unresolved as the previous primary contradiction. Wherever the item is placed after this interaction, it's apparent the team isn't going to be satisfied. However, the team does clear one hurdle: now that the context of the discussion is OUTBOX, options other than wishlist and fire present themselves, and handling the work item as part of a release cycle is discussed.

11.4.2 Addition of new boards as a transformation

The scope of possible options is further expanded in the next daily. A work item that was initially processed as a fire is identified as actually not requiring action. Reference is made to the item's similarity with the one from the previous fragment. The work item depends on an external party, but is not impeded. It is done for a client but the scope is the whole product platform, and it's a technical customer need, but doesn't need to be resolved immediately.

Daily 15, `FIRES`

29:02 "So how many weeks until [the client] need this?" (`D1`)
29:10 "Is this like a non-core, pending for project thing?" (`D1`)
29:28 "Why not move this to [a release cycle]?" (`D2`)
29:48 "Do we have a project for [client]? Can it be there?" (`D3`)
30:28 "We need a box where to put these things. What's the name of the new box?" (`M1`)
"'`MONEYBOX`'" (`D4`)
30:40 "Isn't it kind of priority lane? It's customer needs" (`D5`)
30:45 "How does this differ from any other feature request?" (`D2`)
31:20 "We should have a box, let's call it 'moneybox'" (`M1`)

Now, the direction of the discussion moves to not only rationalisation through existing conceptualisations, but also novel ones. A succession of alternatives are proffered for consideration and shot down with varying amounts of reasoning. Finally, a transformation is achieved by calling out the need for an entirely new place to be putting work items in, thus giving rise to `MONEYBOX`. The secondary contradiction is resolved by the conceptualisation of a new kind of category, allowing the model to better reflect the team's understanding and intent. The act of coming up with a new category itself also serves to redefine the rules of a daily, supporting the use of such a problem solving pattern in the future.

The next fragment presents the team in the act of creating another new place for work items. In doing so, the team frames itself as a stakeholder in the equation of work organisation, and as being capable of declaring things being worthy of doing, of collecting work items to backlogs. Compared to the previous state of affairs this is an empowerment and an expansion of the team's capabilities. The team is beginning to graduate from yes/no, now/never classifier of input to a more nuance-capable actor.

Daily 16, `OUTBOX`

37:50 "Do we some kind of list or ability to make a list of stuff that is not in [part of the platform] but should be?" (`M1`)
"No, we don't have a list" (`D1`, dejected?)

38:06 "Do we even know what these things [that are not part of the platform but should be] are?" (`M1`)

39:15 "So where's the project?" (`M1`)
"Would you like me to make one now?" (`D1`)
"Wouldn't it make sense?" (`M1`)

39:35 "Can we have like 'module core' or something like that?" (`D2`)

We notice in this interaction that the team has resolved a latent contradiction which could be observed at a distance but did not come up in the discussion had in the daily `09` fragment. At that point in time, the team was not an entity that could 'need' something to be done, and the inability to model such a state of affairs reflected that fact. But now the question is, "wouldn't it make sense" to create a new thing to help document the state of need experienced by the team. The contradiction resolved is the fact of being unable to reflect that state of need in the model, and the resolution is to use the team's newfound powers to extend the model accordingly and create `MODULE CORE`.

11.4.3 Facing the effects of new boards

It is after this point that the team needs to face the practical effects of having created new boxes. To have any kind of effect on outcomes, they need to be *used* in some way, and thus the team browses through `MONEYBOX` as part of the daily. The discussion is audibly confused, but uncharacteristically animated with people providing suggestions as almost interjections, as the team looks

to interpret what it sees.

Daily 17, `MONEYBOX`

43:35 "`MONEYBOX`!" (`D1`)
43:43 "These are old things" (`M1`)
43:46 "Let's shovel them somewhere [so] it says 'old things'" (`D2`)
43:49 "Can we put the label..." (`D1`)
"'Longstanding'?" (`D2`)
43:52 "Waiting for money? Money required?" (`D1`)
44:05 "We need an inbox here" (`D2`)

It's found that the `MONEYBOX` needs to be further refined to support the team's use of it in the daily. Recognizing this tertiary contradiction between the new part of practice and the existing practice of effectively going through work items, the team promptly resolves the problem by suggesting that new status labels be applied.

The next fragment finds the team rejecting a work item as reflecting an overly short-sighted approach. The team again doesn't seem to feel they have the right kind of board at hand for the work item. The 'new rules' of problem solving introduced in daily `16` also gradually start to set in: team members see new boxes as an option to take for inbox items.

Daily 18, `INBOX`

25:45 "You know better what needs to be done, but we need to be building a platform" (`M1`)
25:55 "So... we put this in the module backlog, but under a different description... or, wishlist, or..?" (`D1`, unsure which action to take as secretary)
26:10 "What do we do?" (`D1`)
"I don't know what we should do–" (`M1`)

26:25 "Hey, this is a `MODULE CORE` thing, right?" (`D1`)

The situation surfaces a primary contradiction in that there appears to be a conflict between the need for holistic, long-term thinking and quick, one-off fixes. This is resolved by rejecting the initial plan of a one-off fix and rephrasing the related work item. A secondary contradiction also appears when the team is unable to find a place for the rephrased work item. The previously created `MODULE CORE` board is appropriated for this purpose, reconceptualizing it as being capable of admitting the work item. With this transformation in place the team continues satisfied with the outcome.

Another need related to longer term thinking presents itself in the same daily. The team identifies a work item that they identify as needing attention but which is not critical and which may not be taken action with at the moment. This time, the discussion takes a quick leap to the pattern of solutions applied before: would a new box be needed?

Daily 18, `OUTBOX`

35:46 "So this is done?" (`M1`)
"This is still an issue for this single app. Should we have some kind of [..] box?" (`D1`)
35:57 "Or do we have a project [..]?" (`D1`)
36:17 "Should we have a `WATBOX`? Because we have so many things that [..]" (`M1`)
36:36 "That we can leave there and go them through like once a month. `WATBOX`, next to `MONEYBOX`." (`M1`)

The double bind of this situation can be described as follows: There's a need for following up on certain work items after an indeterminate amount of time. But if these items are left in `OUTBOX`, the team has to deal with them (*e.g.* by skipping) in every daily, which is undesirable. Proceeding to a secondary contradiction – on the other hand, if the items are not seen during the course

of a daily, how would action ever be taken? The resolution is to not only have a `WATBOX`, but do decide on what is evidently a different treatment of that box as opposed to others so far.

11.4.4 Newly created boards solidify as accepted practice

The mere existence of a board affords the act of putting work items on it. That act, in the context of a daily, is collaborative and requires the development of a common understanding. The meaning of a board for the team comes into existence in a negotiation, where rationale for the use of a board is presented and either rejected or accepted. An example of such a negotiation occurs in daily `19`, when the placement of a work item in the `MONEYBOX` is challenged.

Daily 19, `INBOX`

15:50 "We don't have a use case" (`M1`)

15:55 "Let's discuss this later" (`D1`)
"Put it to the `MONEYBOX`" (`D2`)
"Or `OUTBOX` if you want to continue the discussion" (`D3`)

16:02 "I don't think this is a `MONEYBOX` issue either" (`M1`)

16:07 "I thought the `MONEYBOX` function was that ... [there] are cases already" (`QA`)

16:13 "No, but there have been indications already, this is an imaginary feature, and now we are waiting for somebody to say 'ok, we will pay x amount'–" (`M1`)

The contradiction is a secondary one, having to do with a recently introduced board and whether a new work item under discussion fits there. A transformation of the concept of `MONEYBOX` takes place, as a rationalization is presented and accepted – the situation is resolved by the work item becoming an example of an acceptable thing to be placing in the `MONEYBOX`.

A lot of noteworthy interactions happen in this very same daily. Next we'll find the team coming to grips with the use of a `MARKETING BOX`, introduced from the outside. The introduction bears relation to the previous agency-taking behavior of this team observed in daily `16`. The board itself belongs to a category of boards where stakeholdership of a work item is the determining factor: who is it that cares about the completion of this item?

Daily 19, `INBOX`

19:36 "'Marketing will go through all the items in the wishlist and put them in... [`MARKETING BOX`]'?" (`D1` reading out loud)

19:42 "Yes. So, we have this very very long wishlist. [...]" (`M1`)

20:05 "It's there, below `QUESTIONS`" (`M1`)

"[Items in the wishlist] get a top 10 priority and the rest is still wishlist. So, just an announcement. [...]" (`M1`)

20:39 "So, `OUTBOX`, longstanding" (`D2`)

"Yeah, we need to see that this is ... [done]" (`M1`)

From the perspective of the other team, this interaction could reflect a quaternary contradiction between their new practice and this team's existing ones. We see that the team cares about whether the new practice starts working, and reflects this intent by modelling the situation as a work item in `OUTBOX`. The latter act may be taken as a step toward realigning the team's orientation and thus resolving the quaternary contradiction.

The themes of long-term planning and wishlists segmented by different stakeholders come together in the following fragment. The team identifies a step in the company's service delivery process that results in significant overhead and thus waste. However, the issue is not so serious as to be a fire, and the team believes a product owner prioritising a wishlist would not rate the problem as high as the team. How should the situation be modelled, if the desire is nevertheless to get the issue solved at the opportune time? The

marketing team has their own wishlist - perhaps the tech team deserves the same treatment!

Daily 19, `INBOX`

23:05 "What if we implement [improvement] [...] it's a huge overhead [having to do without]" (`M1`, motivates desire to act on problem)

23:19 "Yes, huge amounts of waste" (`D1`, agreeing that there's a problem)

23:44 "If it's not a fire then what [..] is it and [when is it] done if ever..?" (`M1`, putting to question how the development of a solution could be modelled)

23:56 "Could we have a state between these cycles and fires?" (`M1`, suggesting a direction of improvement)

"No..! *(laughter)* No." (`D2`)

"Because currently if it's not a fire and it's not-" (`M1`, restating dichotomy in current model)

24:08 "We could have a cloud continuum. But no-one would ever move anything in there." (`D3`, pointing out the need for a new kind of modelling)

"Exactly. This kind of state that would be... It would be like the top ten of [our] input. Like marketing [..]" (`M1`, drawing parallel to `MARKETING BOX`)

24:27 "Isn't priority lane..?" (`QA`, invoking another parallel)

"Sure, but priotity lane means that we are doing it" (`M1`, motivating difference)

25:00 "I made a wishing well for us, and ... this can go there." (`D1`)

"To.. where?" (`M1`)

"The tech wishing well that is right below marketing" (`M1`)

25:42 "Ok, this was a good improvement." (`M1`)

The contradiction underlying this exchange lies in the conflicting needs of different stakeholders and the task of work item priority allocation to reconcile these differences. This is brought up through the identification of a wasteful process and the realisation of an inability to model the team's desire of completing the work item relating to improving that process. The team accepts that this might be resolved by improving their capability to model the situation, which may be taken as the identificiation of a secondary contradiction. As a transformation step, the team constructs the concept of a `TECH WISHING WELL`, taking parallels from `MARKETING BOX`.

The new `TECH WISHING WELL` board is quickly transformed into an accepted part of practice. The team finds that there are other new work items that might best be placed there.

Daily 19, `INBOX`

26:51 "This [defect] is horrible. So could we instead of shoveling this to wishlist .. put this to the newly created wishing well?" (`M1`, breaking from previous norm)

27:42 "So, when do we do this? Wishlist?" (`D1`, offering a resolution for a different issue)
"Let's put this to the wishing well. Not wishlist." (`M1`, suggests again a break from norm)
"Isn't this a wishlist thing?" (`D2`, challenges suggestion)
"We are working on those right now, the push notifications" (`M1`, asserting wishlist is not the right board)

28:46 "Can we put this to wishing well?" (`M1`)
(laughter)
"Let's see if everything gets put to [wishing well], let's see how this works out!" (`M1`)

28:55 "I think [..] everything *we* create is going to wishing well" (`QA`)

29:00 "Exactly [...] this is how the world works" (M1)

The practice of placing work items in the TECH WISHING WELL during INBOX is clearly in conflict with the wishlist-fire dichotomy that still comes up in the team members' questions, presenting a tertiary contradiction. This contradictory state is seemingly accepted with no other rationale besides desire to see what happens. Members remark that the lack of rationale will possibly lead to *everything* being placed in the TECH WISHING WELL as if it was the old wishlist by another name, with the implication that this would be an undesirable outcome. No explicit transformation follows, the contradiction remains unresolved.

At this point in time, the team has a total of four new boards to use as boxes to shovel inbox items to: MONEYBOX, WATBOX, MARKETING BOX and TECH WISHING WELL. In daily 17, the team found they needed more structure for MONEYBOX to use it more effectively. In the following, the team makes additional findings regarding all of these mentioned boxes during a short "drive-through" by the driver and secretary.

Daily 19

45:33 MONEYBOX

"MONEYBOX! ... Nothing new." (D1, as secretary)

45:37 WATBOX

"WATBOX!" (D1)

"WATBOX doesn't need to [be gone] through" (M1)

45:44 "Maybe we should move WATBOX out" (D1)

46:20 MARKETING BOX

"Is marketing already updated?" (M1 rhetorically to driver)

"Yep, it's just my placeholders..." (M1 pointing out there's no action to take)

46:33 NON-CORE MODULE CONTINUUM

"Should we move TECH WISHING WELL elsewhere as well, if

> we don't want to go through it here? ... Moving it next to `WATBOX`." (`D1`)

The boxes come up during the regular course of the daily, but there is nothing to act upon. There's a primary contradiction between the need to spend less collective time on the daily and the thoroughness of inspecting and grooming boards, which is seen as dissatisfaction when a board is inspected but seemingly requires no action from the team. The addition of new boards has exacerbated this condition, bringing about a tertiary contradiction, which is resolved by the team by moving the virtual boards out of the way of the path the driver usually takes when facilitating the daily.

The final steps of the transformational journey occur in daily `20`, when the `TECH WISHING WELL` gets more things assigned on two distinct occasions. We can observe that the team members are now prepared to argue for and rationalise their choice of `TECH WISHING WELL` as the category for a specific work item.

Daily 20, `INBOX`

08:40 "This [work item goes to the] tech wishing well?" (`D1`)
"Yeah, because that was the middle stage between wishlist and fire" (`M1`)

19:27 "So... this goes into wishlist I guess?" (`D2`, accompanied by collective sighing)

19:38 "But isn't this about having centralized signout or something?" (`M1`, providing rationale against wishlist placement)
"Yes but there's no business value [for implementing this in a specific way] so this should go to wishlist..." (`D2`, providing counter-rationale and restating suggestion)
"Maybe this goes somewhere like the tech wishing well?" (`D3`, suggesting a better solution given the rationale)

Two new work items are processed which are considered to belong to `TECH WISHING WELL` board. The team discusses criteria for such items: "middle stage between wishlist and fire" and "no business value". With the presentation and acceptance of these rationale, the meaning of `TECH WISHING WELL` is collaboratively constructed and affirmed. The tertiary contradiction between the previous state of practice and this new one is thus resolved and the use of the board may be seen to have become part of accepted practice for the team.

11.4.5 Transformation summary

At the outset, the team had a way of dealing with `INBOX` items by segmenting them into urgent and non-urgent items. The first would be dealt with immediately, the latter shoveled to wishlists which were to be handled ultimately with the means of a practice entirely separate from the daily. It might be argued that the team was reluctant to make any kind of change due to these wishlists interfacing which other practices, which the daily and the team by themselves aren't empowered to change.

During the transformational journey the team arrives at the understanding that there are different levels of urgencies; different sets of priorities between stakeholders, including themselves; and ways of modelling both of these which are distinct from what the current modelling agrees with or makes possible to express.

Chapter IV

Conclusions

Chapters II and III presented the theoretical framework and empirical material for this study. In this final chapter, conclusions are presented.

Section 12 interprets material from chapter III to provide answers to empirical research questions from Chapter II and the research problem from chapter I. Implications of the study are discussed from practical and theoretical perspectives in section 13. The thesis is concluded in section 14 with an evaluation of the study.

12 Results

In this section, answers to empirical research questions are presented, the research problem of this thesis is answered, and the outcomes are related to the theoretical framework. A conclusion is offered to summarize the results of this thesis.

12.1 Answers to the empirical research questions

> **ERQ1**: How is the observed daily positioned with regard to descriptions of Scrum daily and kanban systems in the theory?

A daily in the Scrum methodology, as described in 5.3, is a short, all-hands meeting held every day. The context of a daily is the ongoing development iteration, or sprint, and the topic will be predefined by the sprint's agenda. The daily's function is defined by the responsibilities of each team member attending: to discuss progress of work items and to communicate any potential impediments. The goal is thus to "keep the ball moving", focus on the essentials of what needs to be done and empower the team members to support each other in the process.

The observed practice is a daily, but it is decidedly not a Scrum one. On the very surface of things, with an average duration of 35 minutes (10.2.2) the daily is significantly longer than the 15 minutes recommended for Scrum. Furthermore, the context of the daily is not a sprint with a fixed agenda, but instead a collection of interrelated, ongoing development processes. Some of these processes are best described as cycles or iterations with a predefined start and conclusion, but not all. The daily was segmented by the use of kanban boards that were heterogenous in structure, which reflected the team's pluralistic understanding of their work.

In terms of absolute time expenditure by segment, we can conclude that the daily has the following main functions: accounting for new information (in the `INBOX` segment), addressing urgent, non-planned work (`FIRES` segment), relating the general status of work, scheduling and impediments (`INTRO`), and keeping track of "discussions", such as improvement of workflows (`OUTBOX`). Secondary functions, which occurred more rarely but on which significant time was spent per occurrence, consist of development cycle reviews and planning (`BOARD OVERVIEW`, `CYCLES`). Auxiliary functions would include keeping track

of non-software-development work, work required for customer relations, and urgent but non-critical work.

According to the literature presented in section 6, kanban systems are a visual work coordination device which allow for inspecting the workflow, or the process through which work items pass through the system. Its purpose is to allow those completing the work to identify the work to be done and act on this information swiftly. This speeds up delivery and enhances learning, while allowing identification of waste throughout the process. A kanban system is intended to be improved upon in a process of continuous improvement, which is collaborative in nature to avoid suboptimization. The use of kanban requires supporting practices, such as those from agile methodologies.

Outside the daily, the team members use the work items represented in their virtual kanban boards to determine what they should be working on at any given time. In section 11.1 we observed that the kanban boards encode relevant workflows when they are understood, such as software development work. As we saw in section 11.2, the layout and structure of the boards is linked to how the daily plays out and is adjusted in the daily. Critically, the members' understanding of *how* the boards are used to communicate what there is to be done is shaped during the daily. The team applies the agile practice of a daily to support their collaborative use of a kanban system.

The team is careful to limit their work in progress as much as they can, but realise that there is some work that is urgent enough to be expedited even at the cost of increasing the amount of things being worked on concurrently. The team's use of the `INBOX` is a mechanism for segregating possible work items into those that can be scheduled for later and those that must be reacted to immediately. Because the team continuously handles any urgent but unplanned work in the `FIRES` segment, they are well equipped for being able to see a category of waste that might be called "rework"; they are aware of their past mistakes and the costs incurred due to addressing them. Finally, the team has a habit of being explicit about their process and its collaborative

improvement. The `OUTBOX` segment serves as a means of keeping track of drives for improvement as well as explicating any outcomes thereof.

In summary, the use of a kanban system with the practice of a daily allows the team to apply virtually all of the kanban principles, which taken together support the lean goal of continuous improvement. The daily also allows the team to accept feedback and adjust their priorities on work to be completed on a day-to-day basis. This distinction from the Scrum daily may be seen as significant, expanding the scope of activity from "progress of predefined work items" to "redefining work and its priorities". It comes at a cost, however, in that the observed dailies were a significant expenditure in terms of the development team's time.

> **ERQ2**: How do contradictions and their transformations manifest in the observed dailies?

Activity theory was presented in section 7 as a way to study innovation in social interactions. According to the activity theoretical viewpoint, innovation happens in practices described as activity systems. Activity systems inherently feature structural tensions that may be resolved through a process of expansive learning. Expansive learning may be taken as a progression of constructing contradictory conditions in a practice and resolving them through transformations of that practice. Describing a practice as an activity system and investigating both contradictions and their transformations in the practice is a way to observe the kind of continuous improvement process introduced in section 6.3.

Deriving from the answer to **ERQ1**, the observed daily may be described as an activity system which exists to make sense of the work that is to be done. The object of activity is the collection of kanban boards and work items therein, and thus the sense-making is achieved by means of inspecting and manipulating the boards and the items. The daily is multi-voiced in terms of bringing together manager, developer and quality assurance perspectives.

Historicity is featured in both the participants' understanding of what happens in a daily carrying over from one daily to the next, as well as in hints of the daily structure being encoded in the boards.

Disturbances, according to section 7.2, are deviations from standard script or breaks of habit and indicate change potential in an activity system. By paying attention to apparent disturbances in the dailies, expressions of contradictions in the activity system could be found and analyzed. In some cases the team could take advantage of these expressions and make changes to the daily so that the contradictions were resolved. An unaddressed contradiction means that the team could not harness the change potential and improve; it could also be interpreted to result from a change that has been found not to be an improvement after all. Changes to the practice that propagate through four tiers of contradiction constitute an expansive learning cycle and become part of new accepted practice.

Section 11.3 identified contradictions exemplifying the daily as relating to effective use of collective time in the daily or outside (primary contradiction), matching work items to suitable boards (secondary), coming to agreement on the implications of a modelling (tertiary), and cross-team coordination (quaternary). Typical to the analysed disturbances was discontent with the modelling: either the model of the situation was experienced as misleading from reality, or the intended course of action was not readily apparent from the model. Addressing the discontent tended to take the form of modifying a single work item, altering a board's structure, deciding on new ways of making work items flow through them or making novel distinctions based on existing constructs.

Transformations tended to be *additive* in nature, not subtractive; the tendency was towards more work items on the boards, more boards for work items and more finely grained process states for items. Subtractive transformations were seen when time constraints for the daily clashed with requirements of modelling, in which case whole segments were discarded from the daily. In

these cases, the boards corresponding to those segments would still be in the model.

Some observed contradictions and transformations could be understood as belonging to sequences of actions in the expansive learning cycle. Section 11.4 describes one cycle where the team identifies a problem with work item categorisation, and creates new boards for reflecting different work item urgencies and stakeholders. As a result, incoming work items are not considered on a binary "fire or not" basis, but a larger spectrum of potential reactions is admitted. This reflects the team collectively having a more detailed understanding of the nature of work they are completing. Finally, the team is able to take a more active role in directing the flows of work items instead of absconding that responsibility.

The progression of disturbances bubbling up, contradictions being expressed, transformations getting suggested and being tried out was continuous. Rare was the daily where not a single transformation step played out. Multiple threads of interrelated transformations could be observed, meaning there were many overlapping cycles of expansive learning ongoing at any time.

In the observed dailies, discovering latent contradictions relating to the kanban model and its use, accompanied with transformations to the same, were identified as the engine to improving the practice. The daily practice allowed these actions to be taken in a manner that can most aptly be described as continuous. Contradictions and transformations therefore manifested as the social mechanism enabling continuous improvement to take place.

12.2 Answer to the research problem

RP: How does a daily support continuous improvement?

The practice of a daily in conjunction with a kanban system supports continuous improvement in a software development team by providing a structure in which the kanban system is observed, new input can be considered and waste may be identified collaboratively, and by serving as a context in which failures in the practice may be brought up and through a process of expansive learning help the team transform the practice.

The theoretical framework describes continuous improvement in practice as a process of collaborative and continuous innovation. Innovation is understood as successive surfacing and resolution of contradictions in a practice by transforming the team's object of activity and thus the practice itself. In the empirical study, the practice of a daily was established and disturbances in the practice were observed, linking them to transformations in the object of activity that the team encoded as changes to the kanban system.

12.3 Discussion

The results of this thesis lie at the intersection of the agile and lean software development discourses. Agile models are described as well documented and readily adaptable (Abrahamsson et al., 2002), but with them being empirically derived it is difficult to say why they work or to modify them successfully (Poppendieck and Poppendieck, 2003, p. xiii-xvi). Lean software development consists of principles that serve as guiding ideas and insights, but are not easy to apply situationally (Poppendieck and Poppendieck, 2003), and even when concrete methods such as kanban are suggested they require supporting methods and significant effort to take up (Ahmad et al., 2013).

Poppendieck and Poppendieck (2003) describe creation of software as inherently a learning activity. However, these claims are made from the viewpoint of and to the benefit of a value-producing system whose output is software. For example, kanban espouses 'collaborative improvement' (Anderson, 2010) because it helps avoid suboptimization (7th lean principle, Poppendieck and Poppendieck, 2003) of the system. The workings of an innovative knowledge community (Hakkarainen et al., 2004), the social context in which the learning and collaborative improvement would happen, is not something lean is positioned to describe.

This thesis suggests the activity theoretical perspective (Engeström et al., 1999) as complementing the agile and lean discourses. In summarization by Hakkarainen et al. (2004), activity theory holds that the source of innovation is overcoming tensions inherent in the activity. A method leveraging this feature is offered by which the daily, a concrete practice at the heart of the agile Scrum methodology (West et al., 2010), can be observed as supporting continuous improvement, a tenet of lean following from the principle of waste reduction (Bhuiyan and Baghel, 2005).

The use of activity theory allowed pointing out *specific features* of the observed daily practice as enabling continuous improvement. Disturbances were felt by team members in social interaction, which allowed surfacing contradictions in the practice. Identifying contradictions afforded transformation of the practice. Transformations didn't always work out to yield direct improvements, but progressions of identified contradictions and attempted transformations could do so. These are ingredients of the social process of innovation, or expansive learning, underlying continuous improvement.

The activity theoretical perspective and vocabulary may be appropriated as a tool in the software development process improvement toolkit. To evaluate the effectiveness of a concrete practice which aims at continuous improvement, practitioners may want to question how the practice affords learning actions to take place. Such implications are discussed further in section 13.1.

12.4 Conclusions

The objective of the thesis was to describe the use of a daily practice in conjunction with the use of a kanban system such that the concept of a daily itself might be expanded upon from dictates of agile software methodology. This was done by searching for evidence that such a use of the daily would support the lean goal of continuous improvement, as the use of a kanban system also does. The theoretical objective was to find a way to describe the daily in such a way as to make possible the identification of mechanisms supporting continuous improvement, which was served by the use of an activity theoretical framing of the daily activities. The empirical study was conducted to further this line of argumentation.

The use of a kanban system could be observed to provide a structure for the daily practice. The object of the daily activity was the model of work expressed in the kanban system, with the objective of grooming new input and collaboratively responding to changing situations and priorities by manipulating the model. An activity theoretical model of expansive learning was offered as an explanation for how continuous improvement occurs. Episodes of disturbance arising from the use of the kanban system in the course of the practice were identified as necessary for the process of improvement. Disturbances brought up contradictions in the model, which made possible transformations in the practice in the form of altering the model or how it was used.

The analysis in this thesis contributes to efforts of devising, implementing and adapting practices based on lean software development, which have the goal of continuous improvement. The results suggest that such practices may benefit from analysis on how they enable the surfacing of disturbances and transformations of the contradictions implied by them.

13 Implications of the study

This section discusses the implications of this study. Implications are divided into practical implications, where views and recommendations on the design of practices on continuous improvement in software development are shared, and theoretical implications which consider directions of future research based on results from this study.

13.1 Practical implications

The nature of findings is in part to challenge received views and informing and improving practice (Robinson et al., 2007). It may be considered unlikely that a receiver of the account of practice elaborated in this study would wish to imitate the practice directly. The researcher hopes that the account may challenge the standard view of both the usefulness of a daily and the application of kanban in software development. Perhaps a receiver will be able to draw inspiration from the claimed benefits of a different kind of daily and be able to use this to structure their own practice; or perhaps they will be reminded of latent opportunities for continuous improvement.

It would be false to claim the observed daily practice as an unqualified success. The study showed that there is value in spending time on a daily, but perhaps the observed daily was still too long, too unfocused, and too neglecting of the value of its participants' time. The fact that participants were observed anxious to move forward from segment to segment could be interpreted as an indicator of boredom or dissatisfaction. The actions taken in dailies also did not have an especially democratic nature, with manager participants taking a large degree of the initiative, which again might be taken antithetical to aims of self-directed action espoused in lean.

Considering the mentioned drawbacks and the obvious situated nature of a specific practice, recommending any of the concrete methods in use in the observed daily would be folly. However, the research points out specific qualities and aims which the observed daily reflected at least to a degree and which would be beneficial for those applying lean principles to keep in mind. The researcher humbly submits the following suggestions for consideration.

Focus on the journey of improvement. Because there is no one right way to do things, the important thing is not to determine such a way and monitor whether your execution is on target. Instead, it is beneficial to design for mechanisms which support change and learning, not enforcing static structures.

Provide an arena for collaboratively inspecting a workflow. A modelling of your team's workflow does little to help the team improve if there is no time and place to interpret and react to what the modelling is telling you. Such a time and place serves the dual purpose of facilitating the creation of collaborative understanding of the work in practice and enabling identification of contradictions. A resulting intersubjective understanding of the current state of practice should help out in being able to collaboratively model transformations of the practice.

Embrace disturbances as indicative of change potential. Foster a culture in which failures, mistakes and contradictory demands may be brought to light as they occur. Structure practices in such a way that they do not break down in the presence of disturbances, so that the disturbances do not need to be extinguished and may instead serve as fuel for collaborative learning.

Expect further contradictions. Resistance to change manifests through contradictions exerted by the part of practice you are trying to change, different parts of the same practice, and other interacting practices. Be ready to address these sources of friction as they come up in order to continue with a fruitful process of transformation. Yielding at the first sign of opposition is

not a recipe for affecting change. Instead, leverage collaborative capability to transform ways of doing and thinking – this is how effective change is achieved, not through a process of imposition which will leave latent contradictions unresolved.

These suggestions are a detachment from the way agile software development might be taught to practitioners. They caution against cargo cult and dogma, asking that a practitioner understand why they are doing what they do and how to go about changing their ways of operating. Yet they also communicate that directions of improvement are fundamentally found in collaborative action, and that the acts of improvement also need to be taken collaboratively. The suggestions offer lean practitioners reminders on constraints and realities which may support development of ways of working.

13.2 Theoretical implications

The theoretical objective of this study was to gain an understanding of how specific practices in software development may be seen as contributive to a process of continuous and collaborative improvement. An activity theoretical perspective of practices was applied to further this goal such that it could not only explain empirical data but serve as a unifying measure in evaluating practices described in the agile and lean discourses of software development.

The research suggests that the use of activity theory to describe practices in the field of software development has explanatory power. We found that the degree to which a practice supported collaborative learning could be analysed by identifying episodes of disturbance which reflected steps in the expansive learning cycle. The activity theoretical perspective therefore offers avenues for further research into software development practices.

The dailies were seen as an arena for expansive learning to take place. In accordance with the research results, expansive learning within lean software development may occur over a time span of days in a continuous process. This is in contrast to the timeboxed "sprint" approach from agile literature, which implies that new input to the process and improvement of the ways of work should happen in interventionary reflection sessions every couple of weeks. If such measures *are* wanted, perhaps their design may be aided by leveraging existing research on interventions from the field of activity theory, such as the ChangeLab method by Engeström et al. (1996).

The research took as a central construct the kanban system used by the team, and the empirical data was interpreted in a way that supports the claim that the system's malleability was an affordance for the team's ability to affect transformations on their practices. The theoretical framework also conceptualised the daily as it exists in agile as a significantly more constrained event, such that it would be unlikely to exhibit the kinds of qualities found in the observed dailies. The strength of this claim could be investigated more thoroughly in an inspection of the occurrence of disturbances in a more traditionally "Scrum" kind of daily, yielding evidence of the degree to which it is the use of a kanban system specifically that enables the team to improve on their practices.

In section 7.1 we delimited the scope of this study to the activity theoretical perspective, in which we consider the process of innovation as that of collaborative contradiction resolution. This was a choice fitting for the investigation of the interactions between daily participants as the focal element, but there is another option. Consider the possibility that we wanted to instead investigate the way the team uses the kanban boards as a knowledge object. In this case we could've done well by picking the *conceptual artefact* based model of knowledge building (Bereiter, 2002), contrasted to Engeström's model by Hakkarainen et al. (2004). A study where the conceptual artefacts are taken in focus could, for instance, look deeper into the team's use of their

tools and find links between the specific ways of using kanban and furthering collaborative learning.

There are evident opportunities in taking the approach from this study further. In the empirical study, fragments of different expansive learning cycles in different phases were observed which suggests that over time, some of these cycles run to completion while some may not. A more longitudinal research could reveal the dynamics of these cycles, which could provide advice on how a daily-like event should be used and structured to optimal benefit and which situational factors affect choices around structuring. Open directions of questioning also include the assumption of a *daily* event, or whether an alternate pacing should be considered, along with whether this choice affects the ways in which disturbances and transformations crop up.

In conclusion, the contribution of this study is to establish the use of activity theory as a valid lens through which to inspect practices in software development. This opens up opportunities in the form of tools in the activity theory toolkit, and practice based research in the large, as being applicable in a software development context. Finally, the study establishes that certain practices in software development may be seen as supporting learning not in individuals but in teams. In other words, the study implies that lean software development practices may support organisational learning.

14 Evaluation

The form of this study is a literature review combined with an empirical case study. This section evaluates the study from perspectives of credibility, transferability, dependability and confirmability, and inherent limitations.

14.1 Credibility, transferability, dependability and confirmability

The research in this thesis encompasses a qualitative (Creswell, 2009) case study (Eisenhardt, 1989) with an ethnographic approach (LeCompte and Goetz, 1982). Lincoln and Guba (1985) suggest that qualitative studies in general be evaluated on criterias of credibility, transferability, dependability and confirmability. LeCompte and Goetz (1982) define the problems of ethnographic research specifically to lie in the capability to establish properties of reliability and validity, which will be linked to the four aforementioned properties in this section.

Credibility is the quality of truthfulness and persuasiveness in causalities and relationships inferred in a study (Lincoln and Guba, 1985). The reasoning in this study followed an abductive inference to the best explanation (Dubois and Gadde, 2002), in which an iterative interplay between theoretical framework and empirical analysis affords the search of a theory with suitable explanatory power. This is consistent with descriptions of case study research process in general (Eisenhardt, 1989) and ethnographical studies specifically (LeCompte and Goetz, 1982).

Transferability is the degree to which findings can be generalized, or whether the findings can be applied in other contexts and to other research subjects (Lincoln and Guba, 1985). In qualitative research, transferability cannot be

determined by the evaluator, but by the receiver (Eisenhardt, 1989; Guba and Lincoln, 1989). In ethnography this determination is supported by the research goals of comparability and translatability (LeCompte and Goetz, 1982). As suggested by LeCompte and Goetz (1982), in this study comparability is supported by delineating the characteristics of the group under observation and the constructs generated in the study such that there exists a basis for comparison for other groups, whereas confident translation is likewise supported by describing research methods, analysis and characteristics of phenomena and groups as explicitly as possible. This is compatible with the requirement by Lincoln and Guba (1985) that the receiver be able to assess applicability to other context by thick description of the research context, theory, methods and findings of the study.

Dependability is defined as the consistency of the study with the capability to provide results independent from the researcher's identity (Guba and Lincoln, 1989). In ethnographical studies, this is taken by LeCompte and Goetz (1982) to be a herculean problem, as the observed phenomena are often unique and the fact of whether the results could be replicated is difficult to establish satisfactorily. LeCompte and Goetz (1982) recommend that the problem be addressed in terms of external and internal reliability. According to the suggestion, this study reports on the researcher's role within the group, delineates the physical context of the study, outlines theoretical premises that shaped the research, and describes data collection strategies used. Internal reliability is enhanced by means of low-inference descriptors, or being as concrete and precise in description and presenting verbatim quotes when possible, which is aided by the use of mechanically recorded data, in this case audio recordings.

Confirmability refers to qualities of neutrality and freedom from bias, values and prejudice (Guba and Lincoln, 1989). Confirmability is justified by ensuring that findings in the study can be traced to the underlying data through an audit trail of inference (Lincoln and Guba, 1985). In this study, confirmability

is afforded by the rich description of empirical study data and findings. The researcher links this property to what LeCompte and Goetz (1982) define as validity, or the the extent to which conclusions represent reality. For purposes of establishing internal validity, this research establishes data that remains stable over the observation period, uses theoretical sampling as a purposive strategy, and accounts for changing group membership and presence over time. External validity is considered by the use of theory to present the observed group's constructs such that they are comparable, and alleviates construct effects with the use of first and third degree data collection methods to validate the understanding of those constructs as being applicable for the group.

14.2 Limitations of the study

This study being ethnographical, the goal was not to generate results generalizable outside the study, rather than to describe the observed phenomena systematically and relate the findings in comparable and translatable fashion (LeCompte and Goetz, 1982). Considering also the activity theoretical perspective taken, Jonassen and Land (2012) warns about the tool being descriptive rather than prescriptive and recommends care to be taken in generalizing the descriptions. There is no expectation that the answers given in the study provide direct solutions to specific problems (Robinson et al., 2007).

The researcher was an established member of the community of practice under investigation, which provided benefits in that the observation situation was less intrusive (Lethbridge et al., 2005) and the researcher could draw on more background to make his interpretations of ongoings (Robinson et al., 2007). Such a position also carries inherent tension (Robinson et al., 2007): if the researcher is part of the community of practice, how do would they be able to consider anything 'strange'? The possibility of an effect by this tension on

the research may only be mitigated, not altogether removed.

The study described the means by which the practice of a daily may support a team's self-directed change of its own practices, which was linked to innovation and further to continuous improvement. However, the linkage provided is only theoretical and, hopefully, plausible for the receiver. An empirically verified causal link from the daily practice to *waste reduction* in the process of software development was not the objective of this study and could not be shown.

Data collection for the study was limited in extensivity, scope, and quality. The duration of the observation period chosen for this study was sufficient to reach conclusions about the existence of expansive learning actions within the practice. The duration was not, however, lengthy enough such that we could have observed cycles of learning that might reasonable be considered 'whole'. Likewise, the scope of the investigation was merely the daily, not software development activities overall. The interplay between daily activities and the practice of software development outside the daily was not investigated. We could not show a causal link from activities in the daily to actual waste reduction elsewhere, only infer that one would exist given the supporting structure. Within the scope of the daily, the audio recordings used as primary data in this study were not by their nature sufficient to fully capture manipulation of work items on kanban boards. Specifically, how the work items discussed by the team moved in the daily was largely unassessable, which limits ability to attain post-hoc understanding of the practice.

Appendix A

Dailies

A1 Daily segment lengths

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
	INBOX	00:15:40	00:13:20

Daily index	Segment	Timestamp	Duration
	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	00:00:41
	NON-FUNCTIONAL CONTINUUM	00:24:46	00:01:19
	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 CONTINUUM	00:07:55	00:05:05
	INBOX	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 MODULE 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 MODULE CONTINUUM	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	00:03:38
	CYCLES	00:03:53	00:12:02
	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 CONTINUUM	00:38:25	00:00:25
	NON-CORE MODULE CONTINUUM	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

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 OUTBOX (ska diskuteras snart)

Capture date	Boards
	QUESTIONS CONTINUUM: Non-functional CONTINUUM: Modules (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 (cycles omitted) 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

Capture date	Boards
	WAT
	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

This section lists outlines of the structures of kanban boards used by the team.

Structure for [FIRE]: This must be emptied, 2015-12-16.

Labels
(unlabeled)
UNHANDLED FIRES INBOX
AWAITING SMOKE TEST (in production)
SMOKETEST FAILED
DEPLOYMENTS (make sure all subtasks are checked)
WAITING IN DEVGYVER FOR MERGE
IMPEDED
IN DEV QA
BACKLOG (waiting to be picked on the table)
WAITING FOR REPRO
LONGSTANDING (extinguished on its own)

Structure for INBOX (Composer 2), 2015-12-16.

Labels
(unlabeled)
INBOX TUTORIAL
Inbox
Needs something
Discuss later
Wishlist inbox
Wishlist
(rest omitted)

Structure for CONTINUUM: Modules, 2015-12-16.

Labels
(unlabeled)
!! NOT AN INBOX !!
SMOKETEST FAILED (in production)
DEPLOYMENTS (make sure all subtasks are checked)
WAITING FOR DEVQA (in devgyver)
DEVQA REJECTED (in devgyver)
WAITING FOR MERGE (in branch)

Structure for OUTBOX (ska diskuteras snart), 2015-12-16.

Labels
(unlabeled)
DONE
TODO
IMPEDED
Longstanding (WIP limit: 18)

Structure for Priority lane, 2015-12-16.

Labels
(unlabeled)
DONE
DOING
TODO
LONGSTANDING

Structure for QUESTIONS, 2015-12-16.

Labels
(unlabeled)
Inbox
Updates asked for
Seen
Badly needs a repro
Client
Plugins
Misc
CLI
Needs repro

Structure for [FIRE] This must be emptied, 2015-12-21.

Labels
(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)

Structure for OUTBOX (ska diskuteras snart), 2015-12-21.

Labels
(unlabeled)
INBOX ABOVE
DONE
ALMOST-BUT-NOT-QUITE-DONE
TODO
IMPEDED
Longstanding (WIP limit: 18)

Structure for OUTBOX (ska diskuteras snart), 2016-01-07.

Labels
(unlabeled)
INBOX ABOVE
DONE
ALMOST-BUT-NOT-QUITE-DONE
TODO
IMPEDED
MISSING IN ACTION

Structure for MONEYBOX, 2016-01-12.

Labels
(unlabeled)
PENDING CUSTOMER ACTION

Structure for OUTBOX (ska diskuteras snart), 2016-01-12.

Labels
(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.

Paul S Adler. The evolving object of software development. *Organization*, 12 (3):401–435, 2005.

M Omair Ahmad, Jouni Markkula, and Markku Oivo. Kanban in software development: A systematic literature review. In *Software Engineering and Advanced Applications (SEAA), 2013 39th EUROMICRO Conference*, pages 9–16. IEEE, 2013.

David J Anderson. *Kanban: successful evolutionary change for your technology business*. Blue Hole Press, 2010.

Paulo Barthelmess and Kenneth M. Anderson. A view of software development environments based on activity theory. *Computer Supported Cooperative Work (CSCW)*, 11(1-2):13–37, 2002.

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`.

Andrew Begel and Nachiappan Nagappan. Usage and perceptions of agile software development in an industrial context: An exploratory study. In

First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007), pages 255–264. IEEE, 2007.

C Bereiter. Education and mind in the knowledge society. *Mahwah NJ*, 2002.

Nadia Bhuiyan and Amit Baghel. An overview of continuous improvement: from the past to the present. *Management Decision*, 43(5):761–771, 2005.

Harry Boer and Frank Gertsen. From continuous improvement to continuous innovation: a (retro)(per) spective. *International Journal of Technology Management*, 26(8):805–827, 2003.

Imelda T Coyne. Sampling in qualitative research. purposeful and theoretical sampling; merging or clear boundaries? *Journal of advanced nursing*, 26 (3):623–630, 1997.

John Creswell. *Research design: Qualitative, quantitative, and mixed methods approaches*. SAGE Publications, Incorporated, 2009.

Anna Dubois and Lars-Erik Gadde. Systematic combining: an abductive approach to case research. *Journal of business research*, 55(7):553–560, 2002.

Kathleen M Eisenhardt. Building theories from case study research. *Academy of management review*, 14(4):532–550, 1989.

Yrjö Engeström. Learning by expanding. an activity-theoretical approach to developmental research. *Orienta-Konsultit Oy*, 1987.

Yrjö Engeström. Activity theory as a framework for analyzing and redesigning work. *Ergonomics*, 43(7):960–974, 2000.

Yrjö Engeström. Expansive learning at work: Toward an activity theoretical reconceptualization. *Journal of education and work*, 14(1):133–156, 2001.

Yrjö Engeström and Annalisa Sannino. Studies of expansive learning: Foundations, findings and future challenges. *Educational research review*, 5(1): 1–24, 2010.

Yrjö Engeström, Jaakko Virkkunen, Merja Helle, Juha Pihlaja, and Ritva Poikela. The change laboratory as a tool for transforming work. *Lifelong Learning in Europe*, 1(2):10–17, 1996.

Yrjö Engeström, Reijo Miettinen, and Raija-Leena Punamäki. *Perspectives on activity theory*. Cambridge University Press, 1999.

Richard P Feynman. Cargo cult science. *The Art and Science of Analog Circuit Design*, page 55, 1998.

Martin Fowler and Jim Highsmith. The agile manifesto. *Software Development*, 9(8):28–35, 2001.

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

Robert L. Glass. Agile versus traditional: Make love, not war! *Journal of Information Technology Management*, 14(12):12–17, 2001.

Egon G Guba and Yvonna S Lincoln. *Fourth generation evaluation*. Sage, 1989.

Kai P Hakkarainen, Tuire Palonen, Sami Paavola, and Erno Lehtinen. *Communities of networked expertise: Professional and educational perspectives*. Elsevier Science, 2004.

J. Highsmith and A. Cockburn. Agile software development: The business of innovation. *IEEE Computer*, 34(9):120–127, 2002. ISSN 0018-9162.

M. Huo, J. Verner, L. Zhu, and M.A. Babar. Software quality and agile methods. In *Proceedings of the 28th Annual International Computer Software and Applications Conference*, 2004.

Marko Ikonen, Petri Kettunen, Nilay Oza, and Pekka Abrahamsson. Exploring the sources of waste in kanban software development projects. In *Software Engineering and Advanced Applications (SEAA), 2010 36th EUROMICRO Conference*, pages 376–381. IEEE, 2010.

David Jonassen and Susan Land. *Theoretical foundations of learning environments*. Routledge, 2012.

Victor Kaptelinin. Activity theory: Implications for human-computer interaction. *Context and consciousness: Activity theory and human-computer interaction*, (1):103–116, 1996.

Hannele Kerosuo, Anu Kajamaa, and Yrjö Engeström. Promoting innovation and learning through change laboratory: An example from finnish health care. *Central European Journal of Public Policy*, 4(1):110–131, 2010.

Mikko Ketokivi and Saku Mantere. Two strategies for inductive reasoning in organizational research. *Academy of Management Review*, 35(2):315–333, 2010.

Philippe Kruchten. Voyage in the agile memeplex. *ACM Queue*, 5(5):38–44, 2007. ISSN 1542-7730.

Kari Kuutti. Activity theory as a potential framework for human-computer interaction research. *Context and consciousness: Activity theory and human-computer interaction*, pages 17–44, 1996.

Margaret D LeCompte and Judith Preissle Goetz. Problems of reliability and validity in ethnographic research. *Review of educational research*, 52(1): 31–60, 1982.

Timothy C Lethbridge, Susan Elliott Sim, and Janice Singer. Studying software engineers: Data collection techniques for software field studies. *Empirical software engineering*, 10(3):311–341, 2005.

Yvonna S Lincoln and Egon G Guba. *Naturalistic inquiry*, volume 75. Sage, 1985.

Lowell Lindstrom and Ron Jeffries. Extreme programming and agile software development methodologies. *Information systems management*, 21(3):41–52, 2004.

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.

Reijo Miettinen. The sources of novelty: A cultural and systemic view of distributed creativity. *Creativity and Innovation Management*, 15(2): 173–181, 2006.

Davide Nicolini. Zooming in and out: Studying practices by switching theoretical lenses and trailing connections. *Organization Studies*, 30(12): 1391–1418, 2009.

Davide Nicolini, Silvia Gherardi, and Dvora Yanow. *Knowing in organizations: A practice-based approach*. ME Sharpe, 2003.

Ikujiro Nonaka and Hirotaka Takeuchi. *The knowledge-creating company: How Japanese companies create the dynamics of innovation*. Oxford university press, 1995.

Leena Norros. System disturbances as springboard for development of operators' expertise. *Cognition and communication at work*, pages 159–176, 1996.

Taiichi Ohno. *Toyota production system: beyond large-scale production*. crc Press, 1988.

Mary Poppendieck and Tom Poppendieck. *Lean Software Development: An Agile Toolkit: An Agile Toolkit*. Addison-Wesley, 2003.

Linda Rising and Norman S Janoff. The scrum software development process for small teams. *IEEE software*, 17(4):26, 2000.

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.

Everett M Rogers and F Floyd Shoemaker. Communication of innovations; a cross-cultural approach. 1971.

Ken Schwaber. Scrum development process. In *OOPSLA Business Object Design and Implementation Workshop*, pages 10–19, 1995.

Ken Schwaber and Jeff Sutherland. The scrum guide. *Scrum Alliance*, 2011. URL `http://www.scrumguides.org/scrum-guide.html`.

Carol Slappendel. Perspectives on innovation in organizations. *Organization Studies*, 17(1):107–129, 1996.

Jeff Sutherland. Agile can scale: Inventing and reinventing scrum in five companies. *Journal of Information Technology Management*, 14(12):5–11, 2001.

Jeff Sutherland and JJ Sutherland. *Scrum: the art of doing twice the work in half the time*. Crown Business, 2014.

Hirotaka Takeuchi and Ikujiro Nonaka. The new new product development game. *Harvard business review*, 64(1):137–146, 1986.

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

Margaret J Wheatley and Myron Kellner-Rogers. Self-organization: The irresistible future of organizing. *Strategy & Leadership*, 24(4):18–24, 1996.

Lisa C Yamagata-Lynch and Michael T Haudenschild. Using activity systems analysis to identify inner contradictions in teacher professional development. *Teaching and Teacher Education*, 25(3):507–517, 2009.

Jason Yip. It's not just standing up: Patterns of daily stand-up meetings. 2006.

Gerald Zaltman, Robert Duncan, and Jonny Holbek. *Innovations and organizations*. John Wiley & Sons, 1973.