Aalto University
School of Science
Degree Programme in Computer Science and Engineering

Eetu Kupiainen

# Agile metrics:
## Why and how

Master's Thesis
Espoo, June 18, 2011

**DRAFT! — February 20, 2014 — DRAFT!**

Supervisor: Prof. Casper Lassenius
Instructors: Juha Itkonen D.Sc. (Tech.)
Mika Mäntylä D.Sc. (Tech.)

Aalto University
School of Science
Degree Programme in Computer Science and Engineering

ABSTRACT OF
MASTER'S THESIS

**Author:** Eetu Kupiainen

**Title:**
Agile metrics: Why and how

| | | | |
|---|---|---|---|
| **Date:** | June 18, 2011 | **Pages:** | 40 |
| **Major:** | Software Engineering | **Code:** | T3003 |
| **Supervisor:** | Prof. Casper Lassenius | | |
| **Instructors:** | Juha Itkonen D.Sc. (Tech.)<br>Mika Mäntylä D.Sc. (Tech.) | | |

Agile development methods are increasing in popularity, yet there are limited studies on the reasons and use of metrics in industrial agile development.

This paper presents results from a systematic literature review. Based on the study, metrics and their use is focused to the following areas: Iteration planning, Iteration tracking, Motivating and improving, Identifying process problems, Pre-release quality, Post-release quality and Changes in processes or tools. The findings are mapped against agile principles and it seems that the use of metrics supports the principles with some deviations.

Surprisingly, we find little evidence of the use of code metrics. Also, we note that there is a lot of evidence on the use of planning and tracking metrics. Finally, the use of metrics to motivate and enforce process improvements as well as applicable quality metrics can be interesting future research topics.

**Keywords:** agile, metrics

**Language:** English

Aalto-yliopisto
Perustieteiden korkeakoulu
Tietotekniikan koulutusohjelma

DIPLOMITYÖN
TIIVISTELMÄ

| | | | |
|---|---|---|---|
| **Tekijä:** | Eetu Kupiainen | | |
| **Työn nimi:** | Ketterät mittarit: Miksi ja miten | | |
| **Päiväys:** | 18. kesäkuuta 2011 | **Sivumäärä:** | 40 |
| **Pääaine:** | Ohjelmistotuotanto ja liiketoiminta | **Koodi:** | T3003 |
| **Valvoja:** | Prof. Casper Lassenius | | |
| **Ohjaajat:** | Tkt Juha Itkonen<br>Tkt Mika Mäntylä | | |

Noh, mistäs teit dippas?

- Nooh, tutkin minkälaisia mittareita ketterät tiimit teollisuudessa käytti, miten ja miksi.

Okei, no mikäs on sen dipan tärkein anti?

- Noh, kategorisoin mittareiden käyttöä ja syitä, johon halukkaat voivat tarkemmin tutustua.

No miksi kukaan haluis noita ihmetellä?

- Mulla (timolla) on sellainen teoria, että on syitä, jotka aiheuttaa ohjelmistoprojektien epäonnistumisia. Mäppäsin löydettyjä mittareita em. syykategorioita vasten, joten voin sanoa että mitä mittareita kannattaisi käyttää jos haluaa ehkäistä tietyistä syistä johtuvia failureita.

- Lisäksi mä näytän miten suurempi osa on reaktiivisia mittareita.

- Lisäksi ideana oli, että pystyisin luomaan jonkinlaisen tärkeiden mittareiden luokittelun... Tämä on osoittautunut aika hankalaksi. Mulla on nyt yksittäisiä mittareita, joita on ylistetty - ja osaan myös sanottu miksi. Ne voin esittää. Lisäksi jos kvantitatiivisesti ajateltuna, näyttää siltä että effort estimaatit on tärkeitä agiilissa, koska niitä näkyi paljon. Toisinpäin ajateltuna, eli mitkä ei oo tärkeitä, niin vois sanoa et koodimittarit. Niitä ei ollut juuri yhtään.

| | |
|---|---|
| **Asiasanat:** | ketterä, mittarit |
| **Kieli:** | Englanti |

# Acknowledgements

I wish to thank all students who use LaTeX for formatting their theses,

Espoo, June 18, 2011

Eetu Kupiainen

# Abbreviations and Acronyms

| | |
|---|---|
| 2k/4k/8k mode | COFDM operation modes |
| 3GPP | 3rd Generation Partnership Project |
| ESP | Encapsulating Security Payload; An IPsec security protocol |
| FLUTE | The File Delivery over Unidirectional Transport protocol |
| e.g. | for example (do not list here this kind of common acronymbs or abbreviations, but only those that are essential for understanding the content of your thesis. |
| note | Note also, that this list is not compulsory, and should be omitted if you have only few abbreviations |

# Contents

# Chapter 1

# Introduction

Software metrics have been studied for decades and several literature reviews have been published. Yet, the literature reviews have been written from an academic viewpoint that typically focuses on the effectiveness of a single metric. For example, Catal et al. review fault prediction metrics (Catal and Diri, 2009), Purao and Vaishnavi (2003) review metrics for object oriented systems and Kitchenham (2010) performs a mapping of most cited software metrics papers. To our knowledge there are no systematic literature reviews on the actual use of software metrics in the industry.

Agile software development is becoming increasing popular in the software industry. The agile approach seems to be contradicting with the traditional metrics approaches. For example, the agile emphasizes working software over measuring progress in terms of intermediate products or documentation, and embracing the change invalidates the traditional approach of tracking progress against pre-made plan. However, at the same time agile software development highlights some measures that should be used, e.g., burndown graphs and 100% automated unit testing coverage. However, measurement research in the context of agile methods remains scarce.

The goal of this paper is to review the literature of actual use of software metrics in the context of agile software development. This study will lay out the current state of metrics usage in industrial agile software development based on literature. Moreover, the study uncovers the reasons for metric usage as well as highlights actions that the use of metrics can trigger.

In this paper, the cover following research questions are covered:

- RQ1: Why are metrics used?
- RQ2: What actions do the use of metrics trigger?
- RQ3: Which metrics are used?

- RQ4: What metrics are important?

## 1.1 Problem statement

## 1.2 Structure of the Thesis

This thesis is structured as follows. Chapter 3 describes how the SR was conducted. Chapter 4 reports the results from the study. Chapter 5 discusses about the findings and how they map to agile principles. Chapter 6 concludes the paper. !Fixme **LALA** Fixme!

# Chapter 2

# Background

## 2.1 Causes for software project failures

## 2.2 Evidence based software engineering

## 2.3 Previous metric research

## 2.4 Aims and research questions

The aim of this paper is to provide

# Chapter 3

# Review method

Systematic review (SR) was chosen as research method because we are trying to understand a problem instead of trying to find a solution to it. Also, there was already existing literature that could be synthesized.

## 3.1 Protocol development

Kitchenham's guide for SRs (Kitchenham, 2004) was used as a basis for developing the review protocol. Additionally, a SR on agile development (Dybå and Dingsøyr, 2008) and a SR on SR (Kitchenham and Brereton, 2013) were used to further understand the challenges and opportunities of SRs. The protocol was also iterated in weekly meetings with the instructors, as well as in a pilot study.

## 3.2 Search and selection process

The strategy for finding primary studies was following:

- Stage 1: Automated search
- Stage 2: Selection based on title and abstract
- Stage 3: Selection based on full text. Conduct data extraction and quality assessment.

Table 3.1 shows the selection funnel in terms of the number of papers after each stage.

Table 3.1: Paper selection funnel

| **Phase** | **Amount of papers** |
|---|---|
| Phase 1 | 774 |
| Phase 2 | 163 |
| Phase 3 | 29 |

Scopus database [1] was used to find the primary documents with automated search. Keywords include popular agile development methods and synonyms for the word metric. The search was improved incrementally in three phases because we noticed some key papers and XP conferences were not found initially. The search strings, hits and dates can be found from appendix A.

The selection of the primary documents was based on an inclusion criteria: *papers that present empirical findings on the industrial use and experiences of metrics in agile context.* The papers were excluded based on multiple criteria, mainly due to not conforming our requirements regarding empirical findings, agile and industrial context, and the quality of the results. Full criteria are listed in appendix B.

In stage 1, Scopus was used as the only search engine as it contained the most relevant databases IEEE and ACM. Also, it was able to find Agile and XP conference papers. Only XP Conference 2013 was searched manually because it couldn't be found through Scopus.

In stage 2, papers were included and excluded by the researcher based on their title and abstract. As the quality of abstracts can be poor in computer science Kitchenham (2004), full texts were also skimmed through in case of unclear abstracts. Unclear cases were discussed with the instructors in weekly meetings and an exclusion rule was documented if necessary.

The validity of the selection process was analysed by performing the selection for a sample of 26 papers also by the second instructor. The level of agreement was substantial with Kappa 0.67 **?**.

Stage 3 included multiple activities in one work flow. Selection by full text was done, data was coded and quality assessment was done. Once again, if there were unclear papers, they were discussed in meetings. Also, selection of 7 papers was conducted by the second instructor with an almost perfect agreement, Kappa 1.0 **?**.

[1] http://www.scopus.com

## 3.3 Data extraction

Integrated coding was selected for data extraction strategy **?**. It provided focus to research questions but flexibility regarding findings. Deductive coding would have been too restraining and inductive coding might have caused too much bias. Integrated coding made it possible to create a sample list of code categories: Why is measurement used?, How is measurement used? and Metrics.

The coding started with the researcher reading the full text and marking interesting quotes with a temporary code. After, reading the full text the researcher checked each quote and coded again with an appropriate code based on the built understanding. In weekly meetings, we slowly built a rule set for collecting metrics:

- Collect metric if team or company uses it.
- Don't collect metrics that are only used for the comparison and selection of development methods.
- Don't collect metrics that are primarily used to compare teams.
- Collect metric only if something is said about why it is used or what actions it causes.
- Collect metric if it is described as important.

Atlas.ti Visual QDA(Qualitative Data Analysis), version 7.1.x was used to collect and synthesize the qualitative data.

To evaluate the repeatability of finding the same metrics, second instructor coded metrics from three papers. Capture-recapture method **?** was then used which showed that 90% of metrics were found.

Table 3.2: Add caption

| Code | Amount of quotations |
|---|---|
| ImportanceRelatedToMetric | 45 |
| How is measurement used? | 61 |
| Why use this metric? | 151 |
| Metrics | 108 |

A quality assessment form adopted from Dybå and Dingsøyr (2008) was used to evaluate the quality of each primary study. Detailed list of quality

assessment questions can be found in appendix C. Additionally, a relevancy factor was added to the same assessment to describe how useful the paper was for this study. The scale for the relevancy factor is:

- 0 = doesn't contain any information regarding metrics and should be already excluded
- 1 = only descriptions of metrics with no additional info
- 2 = some useful information related to metrics
- 3 = a good amount of relevant information regarding metrics and metric usage

## 3.4 Data synthesis

Data synthesis followed the steps recommended by Cruzes et al. **?**. Process started by going through all quotes within one code and giving each quote a more descriptive code describing the quote in high level. Then the descriptive codes were organized in groups based on their similarity. These groups were then given a high level code which are seen as categories in table 4.6.

# Chapter 4

# Results

This chapter presents the preliminary results from the systematic literature review. Table 4.1 shows the distribution of primary documents by publication channels. Table 4.3 lists the distribution of agile methods and table 4.4 lists the distribution of domains.

## 4.1 Quality of studies

The perceived quality of the studies varied a lot (from 0 to 10). Even with many low quality studies they were included since they still provided valuable insight. For example in some cases experience reports can provide more valuable data than a high scoring research papers.

According to the assessment control group and reflexivity had lowest total scores while value for research, context and findings scored the highest.

## 4.2 How and Why

Categories for the reasons and the use of measurements are listed in table 4.6. The following chapters will describe each category in more detail.

### 4.2.1 Iteration planning

Many metrics were used to support iteration planning. The metrics were used for task prioritization and scoping of the iteration.

Many metrics were focused to help in the prioritization of the tasks for the next iteration **???**. Prioritization of features was affected by a metric that measured the amount of revenue a customer is willing to pay for a feature **?**.

Table 4.1: Publication distribution of primary studies

| Publication channel | Type | # | % |
|---|---|---|---|
| Agile Conference | Conference | 8 | 38 |
| HICCS | Conference | 3 | 14 |
| ICSE | Workshop | 2 | 10 |
| XP Conference | Conference | 2 | 10 |
| Agile Development Conference | Conference | 1 | 5 |
| APSEC | Conference | 1 | 5 |
| ASWEC | Conference | 1 | 5 |
| Elektronika ir Elektrotechnika | Journal | 1 | 5 |
| Empirical Software Engineering | Journal | 1 | 5 |
| EUROMICRO | Conference | 1 | 5 |
| ICSE | Conference | 1 | 5 |
| ICSP | Conference | 1 | 5 |
| IST | Journal | 1 | 5 |
| IJPQM | Journal | 1 | 5 |
| JSS | Journal | 1 | 5 |
| PROFES | Conference | 1 | 5 |
| Software - Prac. and Exp. | Journal | 1 | 5 |
| WETSoM | Workshop | 1 | 5 |

Table 4.2: Distribution of research methods

| Research method | Amount |
|---|---|
| Multicase | 2 |
| Experience report | 7 |
| Singlecase | 19 |
| Survey | 1 |

Table 4.3: Distribution of agile methods

| Agile method | Amount |
|---|---|
| Scrum | 15 |
| XP | 7 |
| Lean | 5 |
| Other | 5 |

Table 4.4: Distribution of domains

| Domain | Amount |
|---|---|
| Telecom | 10 |
| Enterprise information system | 7 |
| Web application | 4 |
| Other | 11 |

Effort estimation metrics were used to measure the size of the features **?**. Furthermore, velocity metrics were used to calculate how many features is the team able to complete in an iteration **?**. Knowing the teams' effective available hours was found useful when selecting tasks for an iteration **?**. Velocity metrics were also used to improve the next iteration estimates **?**. In one case, task's start and end date metric was used to point out interdependent tasks in the planning phase **?**.

### 4.2.2 Iteration tracking

Purpose of iteration tracking was to track how the tasks selected for the iteration were performed and that necessary modifications were done to the plan to complete the iteration according to schedule.

Metrics helped in monitoring, identifying problems, and predicting the end result by making it transparent to the stakeholders how the iteration is progressing. **????????**.

Progress metrics included number of completed web pages **?**, story completion percentage **?** and velocity metrics **?**. However, using velocity metrics had also negative effects such as cutting corners in implementing features to maintain velocity with the cost of quality **?**. One qualitative progress metric was product demonstrations with customer **?**. Measuring the completion of tasks enabled selecting incomplete tasks to the next iteration **?**.

When the metrics indicated, during an iteration, that all planned tasks could not be completed, the iteration was rescoped by cutting tasks **???** or adding extra resources **??**.

When there were problems that needed to be fixed, whether they were short or long term, the metrics helped in making decisions to fix them **????**. It was possible to base decisions on data, not only use common sense and experience **?**. Balance of work flow was mentioned as a reason for using metrics in multiple papers **???????**. Progress metrics were used to focus work on tasks that matter the most **?**, avoid partially done work **?**, avoid task switching **?** and polishing of features **?**. Finally, open defects metric

Table 4.5: Add caption

| Study | Research | Aim | Context | R.design | Sampling | Ctrl. Grp | Data coll. | Data a |
|---|---|---|---|---|---|---|---|---|
| Abb | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| And | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| Che | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| dos | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| Dub2005 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| Elss | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| Green | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| Greening | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| Haugen | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| Hodgetts | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| Hodgkins | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| Hong | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| Jakobsen | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| Janus | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Keaveney | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| Mahnic | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| Middleton | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |
| Mujtaba | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| Pet2010SPI | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| Pet2010eff | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| Pet2011 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| Pet2012 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| Polk | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| Seikola | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| Staron2010 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| Staron2011 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| Talby 2006 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| Talby 2009 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| Trapa | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Trimble | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| Tudor | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| Total | 16 | 15 | 20 | 15 | 19 | 7 | 14 | 13 |

Table 4.6: Categories for measurement usage

| Categories | Sources |
|---|---|
| Iteration planning | ????? ???? |
| Iteration tracking | ????? ????? |
| | ????? ?? |
| Motivating and improving | ????? ??? |
| Identifying process problems | ????? ????? |
| Pre-release quality | ???? |
| Post-release quality | ???? |
| Changes in processes or tools | ???????? |

was used to delay a release ?.

### 4.2.3 Motivating and improving

This section describes metrics that were used to motivate people and support team level improvement of working practices and performance.

Metrics were used to communicate different data about the project or product to the team members ?????. Measurement data motivated teams to act and improve their performance?????. Some examples included fixing the build faster by visualizing build status ??, fixing bugs faster by showing amount of defects in monitors ? and increasing testing by measuring product size by automated tests that motivated team to write more tests ?.

Metrics were also used to prevent harmful behaviour such as cherry picking features that are most interesting to the team. Measuring work in progress (WIP) and setting WIP limits prevented cherry picking by enforcing only two features at a time and thus preventing them from working on lower priority but more interesting features.?

### 4.2.4 Identifying process problems

Metrics were often used to identify or avoid problems in processes and work flows. This chapter describes how metrics were used to spot problems.

There were multiple cases highlighting how metrics are used to identify or predict problems in order to solve or avoid them ??????.

Sometimes there were work phases where no value was added, e.g., “waiting for finalization”. This type of activity was called waste and was identified by using lead time. ?

Story implementation flow metric describes how efficiently a developer has been able to complete a story compared to the estimate. This metric helped

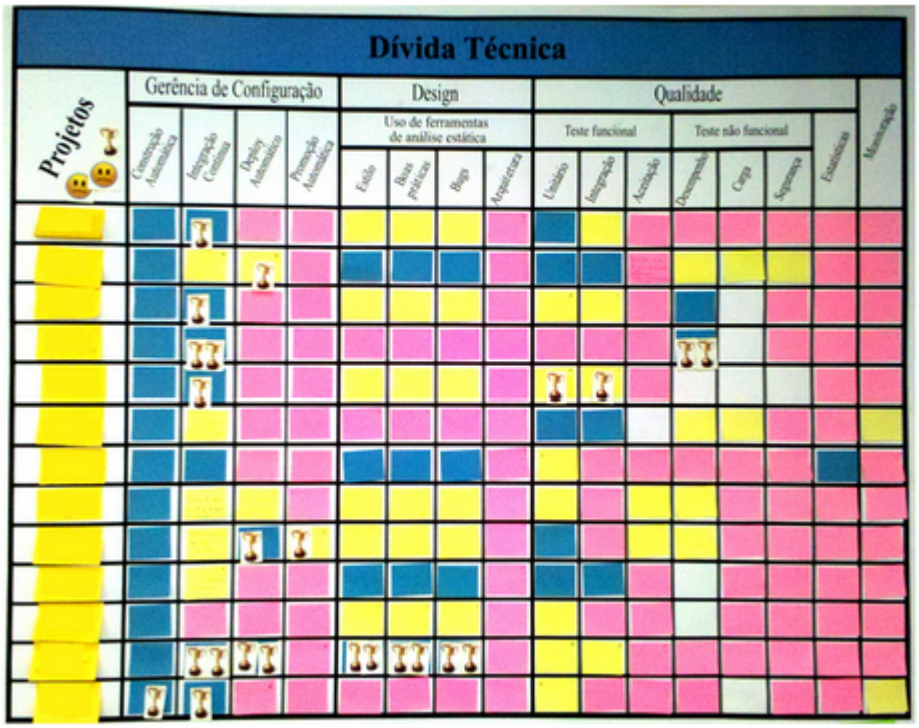


Figure 4.1: Technical debt.

to identify a problem with receiving customer requirement clarifications. ?

Creating awareness with defect trend indicator helped to take actions to avoid problems ?. One common solution to problems was to find the root cause ??.

### 4.2.5 Pre-release quality

Metrics in the pre-release quality category were used to prevent defects reaching customers and to understand what was the current quality of the product.

Integration fails was a problem to avoid with static code check metrics ?. Moreover, metrics were used to make sure that the product is sufficiently tested before the next step in the release path ??. Additionally, making sure that the product is ready for further development was mentioned ?.

Some metrics forced writing tests before the actual code ?. Technical debt was measured with a technical debt board that was used to facilitate discussion on technical debt issues ?.

### 4.2.6 Post-release quality

Metrics in post-release quality deal with evaluating the quality of the product after it has been released.

Customer satisfaction, customer responsiveness, and quality indicators were seen as attributes of post-release quality. Some metrics included customer input to determine post-release quality **???** while other metrics used pre-release data as predictors of post-release quality **???**. Customer related metrics included, e.g., defects sent by customers**?**, change requests from customers **?** and customer's willingness to recommend product to other potential customers **?**. Quality prediction metrics included defect counts **?**, maintenance effort **?** and deferred defect counts **?**.

### 4.2.7 Changes in processes or tools

This chapter describes the reported changes that applying metrics had for processes and tools. The changes include changes in measurement practices, development policies, and the whole development process.

The successful usage of sprint readiness metric and story flow metric changed company policy to have target values for both metrics as well as monthly reporting of both metrics by all projects **?**.

At Ericsson by monitoring the flow of requirements metric they decided to change their implementation flow from push to pull to help them deliver in a more continuous manner. Also, based on the metric they added an intermediate release version to have release quality earlier in the development cycle.**?**

Changes to requirements management were also made based on lead time in other case at Ericsson. Analysing lead time contributed to delaying technical design after purchase order was received, providing customer a rough estimate quickly and merging the step to create solution proposal and technical design. **?**

Problem with broken build, and the long times to fix the build, led to measurements that monitor and visualize the state of the build and the time it takes to fix it **???**.

Also, additional code style rules were added to code check-in and build tools so that builds would fail more often and defects would get caught before release **??**.

Similarly, testing approaches were changed based on flow metrics. Using lead time led to that integration testing could be started parallel to system testing **?**. Also, throughput of a test process showed insufficient capability to handle the incoming features, which led to changing the test approach **?**.

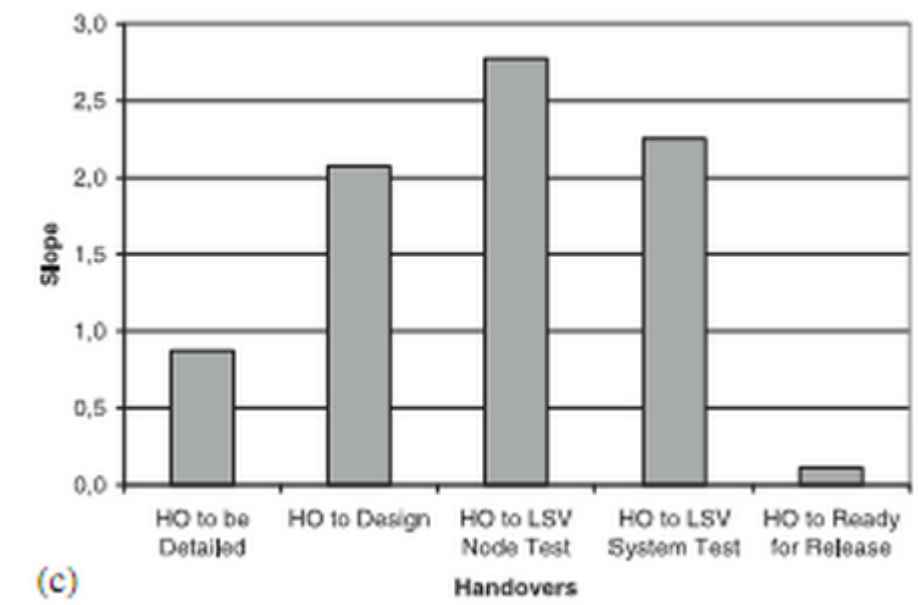


Figure 4.2: Handovers

## 4.3 Metrics & categorization

Metrics are listed by primary study in appendix, see table D.1.

Metrics could help in decreasing software project failures due to prioritization and resource & schedule issues in iteration tracking. Metrics here are all reactive.

Iteration tracking metrics could help in monitoring issues that lead to project failures. Metrics are mostly reactive.

Metrics that are used to motivate and improve people can be used to solve issues related to values & responsibilities and company policies that lead to project failures. Metrics are both reactive and proactive.

Metrics that are used to point identify problems can be used to prevent Method causes for failures. Metrics are almost all proactive.

Metrics that are used to improve or understand pre-release quality can be used to prevent failures that are caused by value & responsibility, task output and existing product related issues. Metrics are mostly reactive.

Metrics that are used for post-release quality can be used to prevent failures that are caused by customers' and users' opinions. Metrics are mostly proactive.

In general, used metrics were more reactive than proactive.

In general, it seems that metrics are used the most to prevent project

Table 4.7: Add caption

| | **My categories** | | | | | | |
|---|---|---|---|---|---|---|---|
| | Iteration planning | Iteration tracking | Motivating and improving | Identifying process problems | Pre-release quality | Post-release quality | C |
| **Causes for failures** | | | | | | | |
| People | | | | | | | |
| Instructions & Experience | 1 | 0 | 0 | 1 | 0 | 0 | |
| Values & responsibilities | 0 | 0 | 2 | 2 | 3 | 0 | |
| Cooperation | 1 | 0 | 0 | 0 | 0 | 0 | |
| Company Policies | 0 | 0 | 3 | 0 | 0 | 0 | |
| Tasks | | | | | | | |
| Task Output | 1 | 0 | 0 | 2 | 4 | 0 | |
| Task Difficulty | 0 | 0 | 0 | 1 | 0 | 0 | |
| Task Priority | 5 | 1 | 0 | 0 | 0 | 0 | |
| Methods | | | | | | | |
| Work Practices | 0 | 0 | 0 | 7 | 2 | 0 | |
| Process | 0 | 1 | 1 | 9 | 0 | 0 | |
| Monitoring | 0 | 10 | 0 | 1 | 4 | 0 | |
| Environment | | | | | | | |
| Existing Product | 0 | 0 | 0 | 0 | 6 | 0 | |
| Resources & Schedules | 11 | 3 | 0 | 0 | 0 | 0 | |
| Tools | 0 | 0 | 0 | 0 | 0 | 0 | |
| Customers & Users | 0 | 1 | 0 | 0 | 0 | 8 | |
| CannotSayCauseCategory | 0 | 2 | 0 | 0 | 0 | 0 | |
| Total | 19 | 18 | 6 | 23 | 19 | 8 | |

failures that are caused by Methods, then by Environment and not so much about People or Tasks.

## 4.4 Important metrics

This section describes metrics that were considered important.

Progress as working code was considered as one of the cornerstones of agile [Trimble20134826].

Story flow percentage and velocity of elaborating features were considered as key metrics for monitoring projects. Also, a minimum 60% value for flow was identified. Similarly, velocity for elaborating features should be as fast as velocity of implementing features. Also, they said using both metrics *"drive behaviors to let teams go twice as fast as they could before"*. [Jakobsen2011168]

Net Promoter Score was said to be *"one of the purest measures of success"* [Green2011].

According to a survey projects that were said to be definitely successful 77% measured customer satisfaction often or always. Also, the more often customer satisfaction would be measured the more likely it would be that the project would have good code quality and the project would succeed. [Abbas]

Story percent complete metric was considered valuable since it embraces test driven development - no progress is made before test is written. Also, percent complete metric is considered more accurate than previous metric. Moreover, it gives normalized measure of progress compared to developer comments about progress. Additionally, story percent complete metric leverages existing unit testing framework and thus requires only minimal overhead to track progress. Team members seemed to be extremely happy about using the metric. [Trapa2006243]

Pseudo-velocity was considered essential for release planning [Polk2011263].

In an agile survey [Abbas] project success had significant positive relationship with team velocity, business value delivered, running testing features, defect count after testing and number of test cases. !FIXME **Tässä nyt puhutaan kuitenkin mittareista mistä ei juurikaan muuta sanota kuin nimi** FIXME!

Effort estimates were consider important in release planning especially in terms of prioritization [Haugen].

Burndown was valuable in meeting sprint commitments [Green2011]. Similary, managers said burndown was important in making decisions and managing multiple teams [Dubinsky200512]. However, developers didn't consider

burndown important [Dubinsky200512].

Top teams at Adobe estimated backlog items with relative effort estimates [Green2011].

Practitioners at Ericsson valued transparency and overview of progress that the metrics were able to provide to the complex product development with parallel activities, namely cost types, rate of requirements over phases and variance in handovers [Petersen2011975].

At another case at Ericsson Value Stream Maps (VSM) were used to visualize problem areas and possible improvements. Practitioners valued how the maps were easy to understand. Metrics that were used to build VSM were lead time, processing time and queue time. [Mujtaba2010139]

Defects deferred was seen as a good predictor of post-release quality because it correlated with issues found by the customers [Green2011].

Defect prediction metrics predicted number of defects in backlog and defect trend indicator were seen important to decision making, and their use continued after the pilot period. Key attributes of the metrics were sufficient accuracy and ease of use. [Staron20113]

Technical debt board that visualized the status of technical debt in categories was considered important because it gave a high level understanding of the problems and it was used to plan actions to remove technical debt. It was proven to be useful in their context. [LNBIP01490121]

The following metrics were consider very useful in agile context: number of unit tests, test coverage, test-growth ratio and broken builds. The benefit for the number of unit tests is not well described except that it provides *"first insights"*. Test coverage provides info on how well the code is tested. Test-growth ratio is useful in projects where old codebase is used as basis for new features. Fixing broken builds prevents defects reaching customers. [Janus20129]

# Chapter 5

# Discussion

## 5.1 Implications for practice

To provide implications to practice we map our findings to the principles of agile software development **?** categorized by Patel & al. **?**. For each paragraph we use the naming by Patel et al. and provide references to the agile practices by numbers.

Communication and Collaboration (principles 4 and 6) was reflected in metrics that motivated a team to act and improve, see section 4.2.3. Also, progress metrics were used to communicate the status of the project to the stakeholders, see section 4.2.2.

Team involvement (5,8) was reflected in metrics that motivated team to act and improve, see section 4.2.3. Also, to promote sustainable development metrics were targeted to balance the flow of work, see section 4.2.2.

Reflection (12) was visible in metrics that were used to identify problems and to change processes, see section 4.2.4 and section 4.2.7.

Frequent delivery of working software (1,3,7) was directly identified in one paper, where the team measured progress by demonstrating the product to the customer **?**. Additionally, there were cases where e.g. completed webpages **?** were the primary progress measure. Also, many metrics focused on progress tracking and timely completion of the iteration, see section 4.2.2. However, some other measures from section 4.2.2 show that instead of working code agile teams followed completed tasks and velocity metrics.

An integral part of the concept of working software is measuring post-release quality, see section 4.2.6. This was measured by customer satisfaction, feedback, and customer defect reports. It was also common to use pre-release data to predict post-release quality. Agile developers tend to measure the end product quality with customer based metrics instead of the traditional

quality models, such as ISO/IEC 25010 (ISO/IEC, 2010).

Managing Changing Requirements (2) was seen in the metrics that support prioritization of features each iteration, see section 4.2.1. Additionally, different metrics helped keeping the internal quality of the product high throughout the development which then provided safe development of modifications from new ideas, see section 4.2.5.

Design (9,10,11) was seen in focus to measuring technical debt and using metrics to enforce writing tests before actual code, see section 4.2.5. Additionally, the status of build was continuously monitored, see section 4.2.7. However, the use of velocity metric had a negative effect on technical quality, see section 4.2.2. Many metrics focused on making sure that the right features were selected for implementation, see section 4.2.1, thus avoiding unnecessary work.

There were also metrics, or their usage, which were not agile in nature. E.g., maintaining velocity by cutting corners in quality instead of dropping features from that iteration **?**. Also, adding people to project to reach a certain date **??** doesn't seem that agile compared to removing tasks. Adding people can have a negative impact to progress, considering the lack of knowledge and training time required. Moreover, the use of dates to plan interdependent tasks is not agile in nature **?**. Instead, interdependencies should be visible in choosing the tasks to appropriate iterations. Also, the use of number of defects to delay a release **?** is against agile thinking as one should rather decrease the scope to avoid such a situation.

Some agile metrics that work well for an agile team, such as tracking progress by automated tests **?**, or measuring the status of the build **?** can turn against the agile principles if used as an external controlling mechanism. The fifth agile principle requires trust in the team, but if the metrics are enforced outside of the team, e.g., from upper management there is a risk that the metrics turn into control mechanisms and the benefits for the team itself suffer.

### 5.1.1 Comparison to prior studies

Only few papers have broadly studied the reasons for software metrics use in the context of agile software development. Hartmann & Dymond **?** also highlight process improvement as one of the reasons for measurement in their agile metrics paper. Also, they emphasize that creation of value should be the primary measure of progress - which was also seen in our study.

Korhonen **?** found in her study that traditional defect metrics could be reused in agile context - if modified. Defect metrics were also used in many of the primary studies.

Kitchenham's mapping study Kitchenham (2010) identified several code metrics in academic literature. However, in our study we found almost no evidence of code metric use in the industrial context. Maybe it is time to re-evaluate the need for code metrics research if industry doesn't seem to use them.

## 5.2 Limitations

The large shares of specific application domains in the primary documents is a threat to external validity. Seven out of 29 studies were from enterprise information systems domain and especially strong was also the share of ten telecom industry studies out of which eight were from the same company, Ericsson. Also, Israeli Air Force was the case organization in three studies.

The threats to reliability in this research include mainly issues related to the reliability of primary study selection and data extraction. The main threat to reliability was having a single researcher performing the study selection and data extraction. It is possible that researcher bias could have had an effect on the results. This threat was mitigated by analysing the reliability of both study selection and data extraction as described in chapter 3.

Due to iterative nature of the coding process, it was challenging to make sure that all previously coded primary documents would get the same treatment, whenever new codes were discovered. In addition, the researcher's coding "sense" developed over time, so it is possible that data extraction accuracy improved during the analysis. In order to mitigate these risks we conducted a pilot study in order to improve the coding scheme, get familiar with the research method, refine the method and tools.

# Chapter 6

# Conclusions

This study presents the results from a systematic literature review from 29 !Fixme **numero** Fixme!primary studies. According to the researcher's knowledge there is no previous systematic reviews of measurement use in the context of industrial agile software development. This study classifies and describes the main measurement types and areas that are reported in empirical studies. This study provides descriptions of how and why metrics are used to support agile software development.!Fixme **lisää important metrics ja metrics categories** Fixme! This study also analyzed how the presented metrics support the twelve principles of Agile Manifesto **?**.

The results indicate that the reasons and use of metrics is focused on the following areas: Iteration planning, Iteration tracking, Motivating and improving, Identifying process problems, Pre-release quality, Post-release quality and Changes in processes or tools.

This paper provides researchers and practitioners with an useful overview of the measurements use in agile context and documented reasonings behind the proposed metrics. This study can be used as a source of relevant sources regarding researchers' interests and contexts.

Finally, this study identified few propositions for future research on measuring in agile software development. First, in the academia lot of emphasis has been given to code metrics yet this study found little evidence of their use. Second, the applicable quality metrics for agile development and the relationship of pre-release quality metrics and post-release quality are important directions of future research. Third, this study found that planning and tracking metrics for iteration were often used indicating a need to focus future research efforts on these areas. Fourth, use of metrics for motivating and enforcing process improvements can be an interesting future research topic.

!Fixme **lisää important metrics ja metric categories** Fixme!

# References


Cagatay Catal and Banu Diri. A systematic review of software fault prediction studies. *Expert Systems with Applications*, 36(4):7346–7354, 2009.

Tore Dybå and Torgeir Dingsøyr. Empirical studies of agile software development: A systematic review. *Information and Software Technology*, 50(910): 833 – 859, 2008. ISSN 0950-5849. doi: http://dx.doi.org/10.1016/j.infsof.2008.01.006. URL `http://www.sciencedirect.com/science/article/pii/S0950584908000256`.

ISO/IEC. Iso/iec 25010 - systems and software engineering - systems and software quality requirements and evaluation (square) - system and software quality models. Technical report, 2010.

Barbara Kitchenham. Procedures for performing systematic reviews. *Keele, UK, Keele University*, 33:2004, 2004.

Barbara Kitchenham. What's up with software metrics? - a preliminary mapping study. *Journal of Systems and Software*, 83(1):37–51, January 2010. ISSN 0164-1212. doi: 10.1016/j.jss.2009.06.041. URL `http://www.sciencedirect.com/science/article/pii/S0164121209001599`.

Barbara Kitchenham and Pearl Brereton. A systematic review of systematic review process research in software engineering. *Information and Software Technology*, 55(12):2049–2075, 2013.

Sandeep Purao and Vijay Vaishnavi. Product metrics for object-oriented systems. *ACM Computing Surveys (CSUR)*, 35(2):191–221, 2003.

# Primary studies

[S1] K

[S2] D

[S3]

[S4]

[S5]

[S6]

[S7]

[S8]

[S9]

[S10]

[S11]

[S12]

[S13]

[S14]

[S15]

[S16]

[S17]

[S18]

[S19]

[S20]

[S21]

[S22]

[S23]

[S29]

[S30]

# Appendix A

# Search strings

The first search string was:

TITLE-ABS-KEY(software AND (agile OR lean OR "crystal method" OR "crystal clear" OR dsdm OR "dynamic systems development method" OR fdd OR "feature driven development" OR "agile unified process" OR "agile modeling" OR scrumban OR kanban OR scrum OR "extreme programming" OR xp) AND (measur* OR metric OR diagnostic OR monitor*)) AND (LIMIT-TO(SUBJAREA, "COMP")) AND (LIMIT-TO(LANGUAGE, "English"))

It found 512 hits 19 September 2013.

The second search string was:

TITLE-ABS-KEY(software AND (agile OR lean OR "crystal method" OR "crystal clear" OR dsdm OR "dynamic systems development method" OR fdd OR "feature driven development" OR "agile unified process" OR "agile modeling" OR scrumban OR kanban OR scrum OR "extreme programming" OR xp) AND (measur* OR metric OR diagnosticOR monitor*)) AND (LIMIT-TO(LANGUAGE, "English")) AND (LIMIT-TO(SUBJAREA, "ENGI")) AND (EXCLUDE (SUBJAREA, "COMP") OR EXCLUDE(SUBJAREA, "PHYS") OR EXCLUDE(SUBJAREA,"MATE") OR EXCLUDE (SUBJAREA, "BUSI") OR EXCLUDE(SUBJAREA, "MATH") OR EXCLUDE(SUBJAREA, "ENVI") OR EXCLUDE (SUBJAREA, "EART") OR EXCLUDE(SUBJAREA, "DECI") OREXCLUDE (SUBJAREA, "ENER"))

It found 220 hits 7 November 2013.

The third search string was:

TITLE-ABS-KEY(software AND (agile OR lean OR "crystal method" OR "crystal clear" OR dsdm OR "dynamic systems development method" OR fdd OR "feature driven development" OR "agile unified process" OR "agile modeling" OR scrumban OR kanban OR scrum OR "extreme programming" OR xp) AND (measur* OR metric OR diagnosticOR monitor*)) AND (LIMIT-TO(LANGUAGE, "English")) AND (LIMIT-TO(SUBJAREA, "BUSI")) AND (EXCLUDE (SUBJAREA, "ENGI") OR EXCLUDE(SUBJAREA, "COMP"))

It found 42 hits 10 December 2013.

# Appendix B

# Inclusion and exclusion criteria

Inclusion criteria

- Papers that present the use and experiences of metrics in an agile industry setting.

Exclusion criteria

- Papers that don't contain empirical data from industry cases.
- Papers that are not in English.
- Papers that don't have agile context. There is evidence of clearly non-agile practices or there is no agile method named. For example, paper mentions agile but case company has only three releases per year.
- Paper is only about one agile practice, which is not related to measuring.
- Papers that don't seem to have any data about metric usage. Similarly, if there are only a few descriptions of metrics but no other info regarding reasons or usage.
- Papers that have serious issues with grammar or vocabulary and therefore it takes considerable effort to understand sentences.
- Papers where the setting is not clear or results cannot be separated by setting, for example surveys where there is data both from academia and industry.
- Papers where the measurements are only used for the research. For example author measures which agile practices correlate with success.

# Appendix C

# Quality assesment questions

1. Is this a research paper?
2. Is there are a clear statement of the aims of the research?
3. Is there an adequate description of the context in which the research was carried out?
4. Was the research design appropriate to address the aims of the research?
5. Was the recruitment strategy appropriate to the aims of the research?
6. Was there a control group with which to compare treatments?
7. Was the data collected in a way that addressed the research issue?
8. Was the data analysis sufficiently rigorous?
9. Has the relationship between researcher and participants been considered adequately?
10. Is there a clear statement of findings?
11. Is the study of value for research or practice?

# Appendix D

# Metric distribution by primary studies

For now, the Aalto logo variants are shown in Figure D.1.

Table D.1: Metrics by primary studies

| ID | Metrics |
|---|---|
| [Abbas] | Business value delivered, customer satisfaction, defect count after testing, number of test cases, running tested features |
| [Anderson] | Velocity |
| ? | Team available hours, team effective hours, critical defects sent by customer, open defects, test failure rate, test success rate, remaining task effort, team effectiveness |
| ? | Technical debt in categories, build status, technical debt in effort |
| ? | Burndown, check-ins per day, number of automated test steps, faults per iteration |
| ? | Velocity, story estimates |
| ? | Burndown, story points, # of open defects, defects found in system test, defects deferred, net promoter score |
| ? | Story points, task effort, velocity, operation's velocity |
| ? | Effort estimate, actual effort |
| ? | # of defects/velocity |
| ? | Revenue per customer |
| ? | Task's expected start and end date, effort estimate, completed web pages, task done |
| ? | Fix time of failed build, story flow percentage, percentage of stories prepared for sprint, velocity of elaborating features, velocity of implementing features |
| ? | Broken build, test coverage, test growth ratio, violations of static code checks, # of unit tests |
| [Keaveney] | Effort estimate, effort estimate, effort estimate, effort estimate |
| ? | Sprint burndown, release burndown, cost performance index, schedule performance index, planned velocity |
| ? | Common tempo time, number of bounce backs, cycle time, work in progress, customer satisfaction(Kano analysis), effort estimate kits, inventory per phase |
| ? | Lead time, processing time, queue time |
| ? | Change requests per requirement, fault slips, implemented vs wasted requirements, maintenance effort, lead time |
| ? | Rate of requirements per phase, variance in handovers, requirement's cost types |
| ? | number of maintenance requests, number of work items per phase, lead time |
| ? | # of faults, fault-slip-through, # of requests from customers, # of requirements per phase, lead time |
| ? | Work in progress, average velocity, cycle time, pseudo velocity |
| ? | Lead time, work in progress, # of days in maintenance, # of days to overdue, reported hours on CSR |


(a) In English


(b) Suomeksi


(c) På svenska

Figure D.1: Aalto logo variants