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 13, 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:**	30
Major:	Software Engineering	**Code:**	T3003
Supervisor:	Prof. Casper Lassenius		
Instructors:	Juha Itkonen D.Sc. (Tech.) Mika Mäntylä D.Sc. (Tech.)		

A dissertation or thesis is a document submitted in support of candidature for a degree or professional qualification presenting the author's research and findings. In some countries/universities, the word thesis or a cognate is used as part of a bachelor's or master's course, while dissertation is normally applied to a doctorate, whilst, in others, the reverse is true.

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:**	30
Pääaine:	Ohjelmistotuotanto ja liiketoiminta	**Koodi:**	T3003
Valvoja:	Prof. Casper Lassenius		
Ohjaajat:	Tkt Juha Itkonen Tkt Mika Mäntylä		
Kivi on materiaali, joka muodostuu mineraaleista ja luokitellaan			
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, we cover the following research questions: Why are metrics used?, What actions do the use of metrics trigger? and Which metrics are used?

This paper 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!

1.1 Problem statement

1.2 Structure of the Thesis

This thesis is organized as follows...

Chapter 2

Background

2.1 Evidence based software engineering

2.2 Previous metric research

2.3 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 **??**.

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 **??**.

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

3.2.1 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 only 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 primarly used to compare teams.
- Collect metric only if something is said about why it is used or what actions it causes.

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.

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

3.2.2 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 4.4.

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 4.2 lists the distribution of agile methods and table 4.3 lists the distribution of domains.

4.1 How and Why

Categories for the reasons and the use of measurements are listed in table 4.4. The following chapters will describe each category in more detail.

4.1.1 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 **?**.

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 **?**.

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 agile methods

Agile method	Amount
Scrum	15
XP	7
Lean	5
Other	5

Table 4.3: Distribution of domains

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

Table 4.4: 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	????????

4.1.2 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 was used to delay a release ?.

4.1.3 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.**?**

4.1.4 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 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 **??**.

4.1.5 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 **?**.

4.1.6 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 **?**.

4.1.7 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 **?**.

4.2 Metrics

4.3 Important metrics

This section describes metrics that were considered important.

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

Capacity as number of features developed in release was considered better than measuring speed, since speed is generally thought as a attribute of humans. Capacity on the otherhand measures the capabilities of an organization.

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!

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].

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

5.0.1 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 4.1.3. Also, progress metrics were used to communicate the status of the project to the stakeholders, see section 4.1.2.

Team involvement (5,8) was reflected in metrics that motivated team to act and improve, see section 4.1.3. Also, to promote sustainable development metrics were targeted to balance the flow of work, see section 4.1.2.

Reflection (12) was visible in metrics that were used to identify problems and to change processes, see section 4.1.4 and section 4.1.7.

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 4.1.2. However, some other measures from section 4.1.2 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 4.1.6. 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 4.1.1. 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 4.1.5.

Design (9,10,11) was seen in focus to measuring technical debt and using metrics to enforce writing tests before actual code, see section 4.1.5. Additionally, the status of build was continuously monitored, see section 4.1.7. However, the use of velocity metric had a negative effect on technical quality, see section 4.1.2. Many metrics focused on making sure that the right features were selected for implementation, see section 4.1.1, 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.

5.0.2 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.

5.0.3 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, August 2008. ISSN 0950-5849. doi: 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] Kirsi Korhonen. Migrating defect management from waterfall to agile software development in a large-scale multi-site organization: A case study. In Pekka a, Michele Marchesi, and Frank Maurer, editors, *Agile Processes in Software Engineering and Extreme Programming*, volume 31 of *Lecture Notes in Business Information Processing*, pages 73–82. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-01852-7. doi: 10.1007/978-3-642-01853-4_10

[S2] D. Hartmann and R. Dymond. Appropriate agile measurement: using metrics and diagnostics to deliver business value. In *Agile Conference, 2006*, pages 6 pp.–134, 2006. doi: 10.1109/AGILE.2006.17

[S3]

[S4]

[S5]

[S6]

[S7]

[S8]

[S9]

[S10]

[S11]

[S12]

[S13]

[S14]

[S15]

[S16]

[S17]

[S18]

[S19]

[S25]

[S26]

[S27]

[S28]

[S29]

[S30]

Appendix A

First appendix

This is the first appendix. You could put some test images or verbose data in an appendix, if there is too much data to fit in the actual text nicely.

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

(a) In English

(b) Suomeksi

(c) På svenska

Figure A.1: Aalto logo variants