Agile metrics: How and why

Eetu Kupiainen
Department of Computer Science and Engineering
Aalto University
FI- 00076 AALTO, FINLAND
eetu.kupiainen@aalto.fi

Mika Mäntylä
Department of Computer Science and Engineering
Aalto University
FI- 00076 AALTO, FINLAND
mika.mantyla@aalto.fi

Juha Itkonen
Department of Computer Science and Engineering
Aalto University
FI- 00076 AALTO, FINLAND
juha.itkonen@aalto.fi

ABSTRACT

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 present preliminary results from a systematic literature review. 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.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*agile metrics*

General Terms

Measurement

Keywords

agile software development, metrics, measurement, systematic literature review

1. INTRODUCTION

From Mika ▶*Kappaleen pointti: No literature reviews of actual metric use*◀ 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 [2], Purao et al. review metrics for object oriented systems [29] and Kitchenham performs a mapping of most cited software metrics papers [18]. To our knowledge there are no systematic literature reviews on the actual use of software metrics in the industry.

From Mika ▶*Kappaleen pointti: Agile on tärkeää eikä metriikkoja tutkittu*◀

From Juha ▶*Yritin konkretisoida trad vs. agile kontrastia*◀ 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. Due to our research goal, we focus this paper on case studies and actual empirical findings excluding theoretical discussion and models lacking empirical validation. In this paper we, cover the following research questions:

1. Why are metrics used?
2. What actions do the use of metrics trigger?
3. Which metrics are used?

This paper is structured as follows. Section 2 describes how the SR was conducted. Section 3 reports the results from the study. Section 4 discusses about the findings and how they map to agile principles. Section 5 concludes the paper and suggests next steps.

2. 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.

2.1 Protocol development

Kitchenham's guide for SRs [17] was used as a basis for developing the review protocol. Additionally, a SR on agile development [7] and a SR on SR [19] were used to further understand the challenges and opportunities of SRs. The protocol was also iterated in weekly meetings with the authors, as well as in a pilot study.

Table 1: Paper selection funnel

Phase	Amount of papers
Phase 1	774
Phase 2	163
Phase 3	29

2.2 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 1 shows the selection funnel in terms of the number of papers after each stage.

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.

From Juha ▶*Laittaisin myös inclusion ja exclusion kriteerit appendixiin ja jättäisin vain tämänkaltaisen lyhyen kuvauksen tänne*◀ 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.

From Juha ▶*Inclusion criteria ja exclusion criteria siirretty -> appendix*◀

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 first author based on their title and abstract. As the quality of abstracts can be poor in computer science [17], full texts were also skimmed through in case of unclear abstracts. Unclear cases were discussed among researchers 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 author. The level of agreement was substantial with Kappa 0.67 [20].

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 author with an almost perfect agreement, Kappa 1.0 [20].

2.3 Data extraction

Integrated coding was selected for data extraction strategy [4]. 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 first author reading the full text and marking interesting quotes with a temporary code. After, reading the full text first author 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, another researcher coded metrics from three papers. Capture-recapture method [30] was then used which showed that 90% of metrics were found.

2.4 Quality assessment

Quality assessment form adopted from [7] was used to evaluate the quality of each primary document. From Juha ▶*relevancy factoria ei käytetä, eikä raportoida tämän paperin tuloksissa, joten poistaisin sen selityksen, kommentoitu pois nyt*◀

2.5 Data synthesis

Data synthesis followed the steps recommended by Cruzes et al. [4]. 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 7.

3. RESULTS

This chapter presents the preliminary results from the systematic literature review. Table 2 shows the distribution of primary documents by publication channels. **??** lists the primary documents and context info.

Categories for reasons and use of measurements are listed in table 7. The following chapters will describe each category in more detail.

3.1 Iteration planning

Many metrics were used to support iteration planning. The metrics were used for task prioritization, estimating the iteration size and team velocity.

Many metrics were focused to help in the prioritization of the tasks for the next iteration [10, 11, 13]. Prioritization of features was affected by a metric that measured the amount of revenue a customer is willing to pay for a feature [13].

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

Table 2: Publication distribution of primary documents

Publication channel	Type	#	%
Agile Conference	Conference	7	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 3: Add caption

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

Table 4: Add caption

Agile method	Amount
Scrum	15
XP	7
Lean	5
Other	5

Table 5: Add caption

Domain	Amount
Telecom	9
Enterprise information system	10
Web application	4
Other	7

Table 6: Metrics by pdoc

ID	Metrics
[3]	Team available hours, team effective hours, critical defects sent by customer, open defects, test failure rate
[5]	Technical debt in categories, build status, technical debt in effort
[6]	Burndown, check-ins per day, number of automated test steps, faults per iteration
[8]	Velocity
[9]	Burndown, story points, # of open defects, defects found in system test, defects deferred, net promoter score
[10]	Story points, task effort, velocity
[11]	Effort estimate, actual effort
[12]	# of defects/velocity
[13]	Revenue per customer
[14]	Task expected start and end date, effort estimate, completed web pages, task done
[15]	Fix time of failed build, story flow percentage, percentage of stories prepared for sprint, time to establish project foundation, velocity of elaborating features, velocity of implementing features
[16]	Broken build, test coverage, test growth ratio, violations of static code checks, # of unit tests
[21]	Sprint velocity, release velocity, cost performance index, schedule performance index, planned velocity
[22]	Common tempo time, number of bounce backs, cycle time, work in progress
[23]	Lead time, processing time, queue time
[24]	Change request per requirement, fault slips, implemented vs wasted requirements, maintenance effort
[25]	Rate of requirements per phase, variance in handovers, requirement's cost types
[26]	Cumulative flow of maintenance requests, lead time
[27]	# of faults, fault-slip-through, # of requests from customer, # of requirements per phase
[28]	Work in progress, average velocity, cycle time
[31]	Lead time, work in progress, # of days in maintenance, # of days to overdue, reported hours on CSR
[32]	Adaptability, innovation, productivity, ROI
[33]	Throughput
[34]	Defect trend indicator, # of defects, predicted # of defects
[35]	Burndown, check-ins per day, number of automated test steps
[36]	Burndown, # of new defects, number of written and passed tests, task estimated vs actual time, time reported for overhead activities, check-ins per day
[37]	Story estimate, story complete percentage
[38]	Progress as working code
[39]	Costs, schedule

Table 7: Categories for why and how measurement usage

Categories	Sources
Iteration planning	[8, 28, 3, 10, 14] [21, 11, 13, 13]
Iteration tracking	[25, 36, 21, 6, 14] [5, 9, 8, 22, 37] [38, 12, 34, 31, 28] [10, 27]
Motivating and improving	[37, 35, 15, 5, 3] [28, 34, 36]
Identifying process problems	[25, 37, 22, 15, 34] [21, 24, 32, 23, 39, 26]
Pre-release quality	[16, 16, 6, 37]
Post-release quality	[24, 3, 9, 34]
Changes in processes or tools	[15, 25, 23, 5, 34, 16, 33, 27]

Effort estimation metrics were used to measure the size of the features [8]. Furthermore, velocity metrics were used to calculate how many features is the team able to complete in an iteration [28]. Knowing the teams' effective available hours was found useful when selecting tasks for an iteration [3]. Velocity metrics were also used to improve the next iteration estimates [21]. From Juha ▶*Tämä task-riippuvuuden mallintaminen alku ja loppupäivineen on hyvin epäagiilia, tästä jotain discussioniin*◀ From Eetu ▶*Ehkä ajattelit tyyliin jotain Gant charttihässäkkää, mutta mä en jotenkin näe tätä mitenkään epäagiilina - enemmänkin maalaisjärjen käyttönä. Täytyyhän joku keino olla hahmottaa sitä, jos taskeilla on riippuvuuksia. Vai oliko agiilissa tähän joku ajatusmalli mitä en nyt saa päähäni?*◀ In one case, task's start and end date metric was used to point out interdependent tasks in the planning phase [14].

3.2 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. [25, 36, 21, 6, 14, 9, 37, 38]. Progress metrics included number of completed web pages [14], story completion percentage [37] and velocity metrics [6]. From Juha ▶*velocity metriikan käyttäminen etenemisenseurantaan on epäagiilia->ongelmia - tästä asiaa discussioniin*◀ From Eetu ▶*lisätty*◀ However, using velocity metrics had also negative effects such as cutting corners in implementing features to maintain velocity with the cost of quality [8]. One qualitative progress metric was product demonstrations with customer [38]. Measuring the completion of tasks enabled selecting incomplete tasks to the next iteration [14].

When the metrics indicated, during an iteration, that all planned tasks could not be completed, the iteration was rescoped by cutting tasks [21, 6, 22] or extra resources were added [6, 22].

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 [34, 6, 25, 5]. It was possible to base decisions on data, not only use common sense and experience [35]. Balance of work flow was mentioned as a reason for using metrics in multiple papers [28, 24, 27, 10, 25, 6, 15]. Progress metrics were used to focus work on tasks that matter the most [35], avoid partially done work [31], avoid task switching [31] and polishing of features [35]. From Juha ▶*epäagiili käytäntö, kerro lisää*◀ From Eetu ▶*En näe mitään epäagiilia tässä.*◀ Finally, open defects metric was used to delay a release [12].

3.3 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 [37, 35, 28, 34, 36]. Measurement data motivated teams to act and improve their performance[35, 28, 3, 5, 15]. Some examples included fixing the build faster by visualizing build status [15, 5], fixing bugs faster by showing amount of defects in monitors [3] and increasing testing by measuring product size by automated tests that motivated team to write more tests [35].

Metrics were also used to prevent harmful behaviour such as cherry picking features that are most interesting to the team. From Juha ▶*WIP limit kaipaa paremman selityksen, eihän WIP limit estä poimasta sitä mieluisinta tehtävää, vaikka tarkeämpiäkin olisi listalla...*◀ From Eetu ▶*Mä ymmärrän tän niin, että tiimi ei voi nappailla muita tehtäviä ennen kuin tämänhetkinen on valmis. Ei tuota ole tosin hirveen selkeästi kuvattu alkuperäisessä tekstissäkään: "Teams could only work on a maximum of two features or feature level integration at any one time. This also stopped teams 'cherry picking' features they wanted to develop at expense of the whole product."*◀ Measuring work in progress (WIP) and setting WIP limits prevented cherry picking by enforcing only two features at a time. [22]

3.4 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 [25, 37, 21, 24, 32, 23, 39].

Sometimes there were work phases where no value was added, for example, "waiting for finalization". This type of activity was called waste and was identified by using lead time. [26]

From Juha ▶*Tämä mittari ei ole selkeä, täytyy avata hieman enemmän mikä se on*◀ From Eetu ▶*Avattu nyt hieman lisää.*◀ 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 [15].

Creating awareness with defect trend indicator helped to take actions to avoid problems [34]. One common solution to problems was to find the root cause [15, 22].

3.5 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 [16]. Moreover, metrics were used to make sure that the product is sufficiently tested before the next step in the release path [16][6]. Additionally, making sure that the product is ready for further development was mentioned [9].

Some metrics forced writing tests before the actual code [37]. Technical debt was measured with a technical debt board that was used to facilitate discussion on technical debt issues [5].

3.6 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 [24, 9, 3] while other metrics used pre-release data as predictors of post-release quality [34, 24, 9]. Customer related metrics included, for example, defects sent by customers[3], change requests from customers [24] and customer's willingness to recommend product to other potential customers [9]. Quality prediction metrics included defect counts [24], maintenance effort [34], and deferred defect counts [9].

3.7 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 [15].

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 intermediate release version to have release quality earlier in the development cycle.[27]

From Juha ▶*ehkä hieman enemmän voisi paljastaa*◀ From Eetu ▶*avattu*◀ 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. [23]

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 [5, 15, 16].

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 [15, 16].

From Juha ▶*ei avaudu*◀ From Eetu ▶*avattu*◀ Similarly, testing approaches were changed based on flow metrics. Using lead time led to that integration testing could be started paraller to system testing [23]. Also, throughput of a test process showed insufficient to handle the incoming features which led to changing the test approach [33].

4. DISCUSSION

4.1 Implications for practice

To provide implications to practice we map our findings to the values of agile software development [1].

From Mika ▶*TODO: Pitaa selvasti tuoda esiin mappaus. Principlet jarjestykseen. Viite mista principlet tulee. Joka principlen kohdalta ensin mika sita tukee. Ja sitten jos loydettiin jotain ristiriittaista periaatteen kanssa. Lisaksi principlet jarjestykseen. Ja lopuksi mille principlelle ei ollut mittareita. -> Future work Pitaisiko olla ja millaisia*◀ From Eetu ▶*Done.*◀

Agile principle #1: "Our highest priority is to satisfy the customer through early and continuous delivery of valuable software." was seen in the team measuring progress by demonstrating the product to the customer [38].

Agile principle #2: "Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage." was seen in the metrics that support prioritization of features per iteration, see section 3.1. Additionally, different metrics helped keeping the internal quality of the product high throughout the development which then provided safe development of modifications and new ideas, see section 3.5.

Agile principle #3: "Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale" was seen in many metrics focusing on tracking and timely completion of the iteration, see section 3.2

Agile principle #4:"Business people and developers must work together daily throughout the project." was seen how different metrics were used to share information to all stakeholders about the project, see section 3.3 and section 3.2.

Agile principle #5:"Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done" was reflected in metrics that motivated team to act and improve, see section 3.3.

Agile principle #6:"The most efficient and effective method of conveying information to and within a development team is face-to-face conversation." was seen in [5] where a technical debt board measuring the level of technical debt was used to facilitate face-to-face discussion on technical debt issues, see 3.5.

Agile principle #7:"Working software is the primary measure of progress" was directly identified in one paper, where the team measured progress by demonstrating the product to the customer. Additionally, there were cases where for example completed web-pages [14] were the primary progress measure. However, some other measures from section 3.2 show that instead of working code agile teams followed completed tasks and velocity metrics.

Agile principle #8:"Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely." was followed with metrics targeted to balance the flow of work, see section 3.2.

Agile principle #9:"Continuous attention to technical excellence and good design enhances agility." was seen in focus to measuring technical debt and using metrics to enforce writing tests before actual code, see section 3.5. Additionally, the status of build was continuously monitored, see section 3.7. However, the use of velocity metric had a negative effect on technical quality, see section 3.2.

Agile principle #10:"Simplicity—the art of maximizing the amount of work not done—is essential." was seen from different perspectives: on one hand metrics focused on problem/waste identification, see section 3.4, and on the other hand many metrics focused on making sure that the right features were selected for implementation, see section 3.1.

Agile principle #11:"The best architectures, requirements, and designs emerge from self-organizing teams." was seen in metrics that motivate the team to improve, see section 3.3. Other perspective is that since effort estimation is done by the team, the team is then more motivated to accomplish the goal, see section 3.1.

Agile principle #12: "At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly" was visible in metrics that were used to identify problems and to change processes, see section 3.4 and section 3.7.

From Eetu ▶*Mun mielestä seuraavat mittarit ei oo niin ketteriä, mut en osaa oikeen perustella miksi tai sanoa mitä periaatteita vastaan ne olisi: maintenance effort, cost types, defect amounts(?), defects deferred, revenue per customer(?), time to establish project foundation, test coverage, test growth ratio, cost performance index, schedule performance index. Näitä ei kaikkia ole myöskään kuvattu resultseissa, paitsi taulukossa mainittu.*◀

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 [8]. Also, adding people to project to reach a certain date [6, 22] 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.

While the flow metrics Ericsson have a good target of balancing workflow, they seem (or at least they are presented) complicated to use - meaning that one might need considerable effort generate and analyse the metrics, which doesn't fit to the light-weightness of agile.

Contradictory to #5, Talby et al [viite] enforce writing automated test cases as a measure of progress - so in a way they didn't trust the developers to write the test on their own? Similarly, [viite] measured the status of the build to make developers fix the build faster - again, not trusting them to do it on their own.

4.2 Implications for research

It was interesting to notice that there wasn't many code metrics, only the ones mentioned in [16] even though we feel there are many studies regarding the benefits of code metrics. Maybe there are some practical problems implementing and analysing the data from code metrics?

How to measure unmeasured agile principles...

In general, we think there were many metrics that were targeted for the team - instead of high focus on managerial or upper management reporting metrics. Making metrics visible for the team enables them to independently act and improve without the need of rapid supervision and telling people what to do.

From Eetu ▶*(toinen judu mitä täällä voisi olla niin vertailu perinteisiin tai agiilikirjallisuudessa suositeltuihin) (kolmas judu: Koodimittarit oli aika heikosti edustettuna - vain muutama. Joka on sin?ns? mielenkiintoista koska tutkimuksissa ne on hyvin edustettuna(kai). Mut ilmeisesti tila tulee vastaan ni nää vois unohtaa toistaiseksi*◀

4.3 Limitations

Telecommunications sector is widely represented in this study with eight papers from Ericsson. Also, Israeli Air Force was presented in three papers.

Sometimes it was hard to understand which metrics an author was referring when a "why" was described. Moreover, we had to sometimes assume that when author describes the reasons for using a tool, he would be actually talking about the metrics the tool shows.

Whenever a new coding rule was decided it was hard to make sure that all previously coded primary documents would get the same treatment.

Coding "sense" improved over time so it is possible that some information was not spotted from the primary documents in the beginning of the study.

It is possible that researcher bias could have had an effect on the results. First author has positive mindset towards agile methods, as well as towards certain metrics over others.

5. CONCLUSIONS

This paper presents the preliminary results from a systematic literature review. Results indicate that the use and reasoning of metrics is focused on the following areas:

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

We also map the found metrics to the principles of Agile Manifesto [1].

From Eetu ▶*So what now? What did we learn and what should be done now? Mitä nämä tulokset merkitsee?Ainakin voisin sanoa että ihan mukavasti löyty materiaalia. Jonkin verran löytyi samanlaisia asioita eri papereista. Tietyllä tapaa kiinnostaisi vertailu perinteisiä mittareita vastaan. Onko näillä mittareilla loppupeleissä mitään eroa jo vuosikausia käytössä ollleihin mittareihin nähden? Onko jotain yhdistäviä konteksteja havaittavissa jolloin voisi yleistää havaintoja? Myös lista tärkeiksi koetuista mittareista olisi kiva (tämähän mulla on jo pienellä alulla). Voinko suositella tuloksissa esiteltyjä mittareita?*◀

6. ACKNOWLEDGMENTS

U-QASAR rahoitus?

7. REFERENCES

[1] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, and D. Thomas. Manifesto for agile software development.

[2] C. Catal and B. Diri. A systematic review of software fault prediction studies. *Expert Systems with Applications*, 36(4):7346–7354, 2009.

[3] T.-H. Cheng, S. Jansen, and M. Remmers. Controlling and monitoring agile software development in three dutch product software companies. In *Proceedings of the 2009 ICSE Workshop on Software Development Governance, SDG 2009*, pages 29–35, Vancouver, BC, 2009.

[4] D. Cruzes and T. Dyba. Recommended steps for thematic synthesis in software engineering. In *Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on*, pages 275–284, 2011.

[5] P. S. M. dos Santos, A. Varella, C. R. Dantas, and D. B. ao Borges. Visualizing and managing technical debt in agile development: An experience report. In H. Baumeister and B. Weber, editors, *Agile Processes in Software Engineering and Extreme Programming*, volume 149 of *Lecture Notes in Business Information Processing*, pages 121–134. Springer, 2013.

[6] Y. Dubinsky, D. Talby, O. Hazzan, and A. Keren. Agile metrics at the israeli air force. In *Proceedings - AGILE Confernce 2005*, volume 2005, pages 12–19, Denver, CO, 2005.

[7] T. Dybå and T. Dingsøyr. Empirical studies of agile software development: A systematic review. *Information and Software Technology*, 50(9Ŭ10):833–859, Aug. 2008.

[8] A. Elssamadisy and G. Schalliol. Recognizing and responding to "bad smells" in extreme programming. In *Proceedings - International Conference on Software Engineering*, pages 617–622, Orlando, FL, 2002.

[9] P. Green. Measuring the impact of scrum on product development at adobe systems. In *Proceedings of the Annual Hawaii International Conference on System Sciences*, Koloa, Kauai, HI, 2011.

[10] D. Greening. Enterprise scrum: Scaling scrum to the executive level. In *Proceedings of the Annual Hawaii International Conference on System Sciences*, Koloa, Kauai, HI, 2010.

[11] N. Haugen. An empirical study of using planning poker for user story estimation. In *Proceedings - AGILE Conference, 2006*, volume 2006, pages 23–31, Minneapolis, MN, 2006.

[12] P. Hodgetts. Refactoring the development process: Experiences with the incremental adoption of agile practices. In *Proceedings of the Agile Development Conference, ADC 2004*, pages 106–113, Salt Lake City, UT, 2004.

[13] P. Hodgkins and L. Hohmann. Agile program management: Lessons learned from the verisign managed security services team. In *Proceedings - AGILE 2007*, pages 194–199, Washington, DC, 2007.

[14] N. Hong, J. Yoo, and S. Cha. Customization of scrum methodology for outsourced e-commerce projects. In *Proceedings - Asia-Pacific Software Engineering Conference, APSEC*, pages 310–315, Sydney, NSW, 2010.

[15] C. Jakobsen and T. Poppendieck. Lean as a scrum troubleshooter. In *Proceedings - 2011 Agile Conference, Agile 2011*, pages 168–174, Salt Lake City, UT, 2011.

[16] A. Janus, R. Dumke, A. Schmietendorf, and J. Jager. The 3c approach for agile quality assurance. In *Emerging Trends in Software Metrics (WETSoM), 2012 3rd International Workshop on*, pages 9–13, 2012.

[17] B. Kitchenham. Procedures for performing systematic reviews. *Keele, UK, Keele University*, 33:2004, 2004.

[18] B. Kitchenham. What's up with software metrics? - a preliminary mapping study. *Journal of Systems and Software*, 83(1):37–51, Jan. 2010.

[19] B. Kitchenham and P. Brereton. A systematic review of systematic review process research in software engineering. *Information and Software Technology*, 55(12):2049–2075, 2013.

[20] J. Landis and G. Koch. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174, 1977.

[21] V. Mahnic and N. Zabkar. Measuring progress of scrum-based software projects. *Electronics and Electrical Engineering*, 18(8):73–76, 2012.

[22] P. Middleton, P. Taylor, A. Flaxel, and A. Cookson. Lean principles and techniques for improving the quality and productivity of software development projects: A case study. *International Journal of Productivity and Quality Management*, 2(4):387–403, 2007.

[23] S. b. Mujtaba, R. Feldt, and K. b. Petersen. Waste and lead time reduction in a software product customization process with value stream maps. In *Proceedings of the Australian Software Engineering Conference, ASWEC*, pages 139–148, Auckland, 2010.

[24] K. Petersen and C. Wohlin. The effect of moving from a plan-driven to an incremental software development approach with agile practices: An industrial case study. *Empirical Software Engineering*, 15(6):654–693, 2010.

[25] K. Petersen and C. Wohlin. Measuring the flow in lean software development. *Software - Practice and Experience*, 41(9):975–996, 2011.

[26] K. b. Petersen. A palette of lean indicators to detect waste in software maintenance: A case study. *Lecture Notes in Business Information Processing*, 111 LNBIP:108–122, 2012.

[27] K. b. Petersen and C. Wohlin. Software process improvement through the lean measurement (spi-leam) method. *Journal of Systems and Software*, 83(7):1275–1287, 2010. cited By (since 1996)6.

[28] R. Polk. Agile & kanban in coordination. In *Proceedings - 2011 Agile Conference, Agile 2011*, pages 263–268, Salt Lake City, UT, 2011.

[29] S. Purao and V. Vaishnavi. Product metrics for object-oriented systems. *ACM Computing Surveys (CSUR)*, 35(2):191–221, 2003.

[30] G. Seber. *The Estimation of Animal Abundance and Related Parameters*. Blackburn Press, 2002.

[31] M. Seikola, H.-M. Loisa, and A. Jagos. Kanban implementation in a telecom product maintenance. In *Proceedings - 37th EUROMICRO Conference on Software Engineering and Advanced Applications, SEAA 2011*, pages 321–329, Oulu, 2011.

[32] B. Shen and D. Ju. On the measurement of agility in software process. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 4470 LNCS:25–36, 2007.

[33] M. Staron and W. Meding. Monitoring bottlenecks in agile and lean software development projects - a method and its industrial use. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in*

Bioinformatics), 6759 LNCS:3–16, 2011.
[34] M. Staron, W. Meding, and B. Söderqvist. A method for forecasting defect backlog in large streamline software development projects and its industrial evaluation. *Information and Software Technology*, 52(10):1069–1079, 2010.
[35] D. Talby and Y. Dubinsky. Governance of an agile software project. In *Proceedings of the 2009 ICSE Workshop on Software Development Governance, SDG 2009*, pages 40–45, Vancouver, BC, 2009.
[36] D. Talby, O. Hazzan, Y. Dubinsky, and A. Keren. Reflections on reflection in agile software development. In *Proceedings - AGILE Conference, 2006*, volume 2006, pages 100–110, Minneapolis, MN, 2006.
[37] V. Trapa and S. Rao. T3 - tool for monitoring agile development. In *Proceedings - AGILE Conference, 2006*, volume 2006, pages 243–248, Minneapolis, MN, 2006.
[38] J. Trimble and C. Webster. From traditional, to lean, to agile development: Finding the optimal software engineering cycle. In *Proceedings of the Annual Hawaii International Conference on System Sciences*, pages 4826–4833, Wailea, Maui, HI, 2013.
[39] D. Tudor and G. Walter. Using an agile approach in a large, traditional organization. In *Proceedings - AGILE Conference, 2006*, volume 2006, pages 367–373, Minneapolis, MN, 2006.

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.

B. INCLUSION AND EXCLUSION CRITERIA

Inclusion criteria

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

Exclusion criteria From Juha ▶*Tiivistin exclusion kriteerejä*◀

- 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.