Agile metrics: How and why

Eetu Kupiainen
Aalto University
Konemiehentie 2
Espoo, Finland
eetu.kupiainen@aalto.fi

Mika Mäntylä
Aalto University
Konemiehentie 2
Espoo, Finland
mika.mantyla@aalto.fi

Juha Itkonen
Aalto University
Konemiehentie 2
Espoo, Finland
juha.itkonen@aalto.fi

ABSTRACT

Categories and Subject Descriptors

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

General Terms

Theory

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 [3], Purao et al. review metrics for object oriented systems [31] and Kitchenham performs a mapping of most cited software metrics papers [20]. 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*◀ Agile software development is becoming increasing popular in the software industry. The agile approach seems to be contradicting with the traditional metrics approaches. 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 with 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 industrial agile metric papers. Moreover, the study uncovers the reasons for metrics usage as well as highlights actions that the use of metrics can trigger. Due to our research goal, we are more interested in case studies and actual empirical findings than unproven theories or models.

This paper is structured as follows. Section...

1.1 Aims and research questions

The aim of this paper is to provide preliminary results from a systematic review (SR) on agile metrics. Moreover, we are interested on the industrial use of metrics in agile context.

The performed SR has more research questions but for this paper only the following questions are considered:

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

2. REVIEW METHOD

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[19] was used as a basis for developing the review protocol. Additionally, other guidelines[42], a lessons learned from SRs [2], a SR case[9] and a SR on SR[21] were used to further understand the challenges and opportunities of SRs.

The protocol was also iterated in weekly meetings with the researchers, as well as in a pilot described in 2.2.5.

2.2 Search and selection process

The strategy for finding primary studies was following:

1. Initial check that there are enough papers to conduct SR
2. Stage 1: Automated search
3. Pilot
4. Stage 2: Include and exclude based on title and abstract
5. Stage 3: Include and exclude based on full text. Conduct data extraction and quality assessment.

Table 1 shows the selection funnel in terms of amount papers.

WETSoM '14 Hyderabad, India

Table 1: Paper selection funnel

Phase	Amount of papers
Phase 1	774
Phase 2	163
Phase 3	25

2.2.1 Search strings

Scopus was used to find the primary documents with automated search. 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 19.9.2013.

Then we noticed that some previously found key papers were missing from the hits because they were under sub area "Engineering", thus the second search string:

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 7.11.2013.

Then we found out that previous searches didn't include all XP conferences because they were under sub area "Business", thus the third search:

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 10.12.2013

2.2.2 Inclusion criteria

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

2.2.3 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. Also, if there is 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 understand sentences.
- Papers that refer to another paper where the actual case is discussed.
- Papers that are in academic or semi-academic setting - customer or part of workers are from industry. The reason for this is that it doesn't fully represent industry setting as software development methods are likely enforced by academia.
- Papers where results cannot be separated by setting, for example surveys where there is data both from academia and industry. Similarly exclusion if results cannot be separated by software development method.
- Papers where the setting is not clear. For example only mention of context is "100 junior developers".
- Papers that are full conference proceedings. Individual papers should be already be separated individually.
- Papers that where the measurements are only used for the research. For example author measures which agile practices correlate with success.
- Papers that don't even show measurement usage in a pilot setting. For example method or metric is used to a static industrial data set.
- Papers that are about the same case, from the same author and same research focus, basically the paper format has just changed slightly.

2.2.4 Stage 1 - Automatic search

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.

2.2.5 Pilot

A pilot study was conducted in order to refine the aim of the research and get familiar with the research method. Moreover, it was possible to modify the method and tools.

The plan was to go through 15 papers from the automatic search:

- 5 by top relevance
- 5 by top citations
- 5 by random

The pilot resulted in changing citation manager tool from Zotero to Jabref. Also, selection by title and selection by abstract steps were joined together. The quality assessment checklist was decided on.

2.2.6 Stage 2 - Selection by title and abstract

In stage 2 papers were included and excluded based on their title and abstract. As the quality of abstracts can be poor in computer science [19], full texts were also skimmed through in case of unclear abstracts - especially intro, case description and conclusions was checked briefly. Also, if there were any unclear cases - these were discussed in weekly meetings and a descriptive exclusion rule was created if necessary.

Selection process was also done for a sample of 26 papers by one other researcher - the level of agreement was substantial with Kappa 0.67 [22].

2.2.7 Stage 3 - Selection by full text

Stage 3 included multiple activities in one work flow. Selection by full text was done, interesting data was coded and quality assessment was done. Once again, if there were unclear papers, they were discussed in meetings with other researchers. Sample set of 7 papers was also included/excluded by other researcher, with an almost perfect agreement, Kappa 1.0 [22].

2.3 Data extraction

Integrated coding was selected for data extraction strategy [6]. 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, namely:

- Why is measurement used?
- How is measurement used?
- Metrics with metric name
- Interesting results

The coding started with reading the full text and marking interesting data with a temporary code. After, reading the full text I would check each data pieces and code again with an appropriate code based on the built understanding. Metric codes were named based on the similarity with the text it was collected from.

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

Atlas.ti was used to collect the qualitative data(viiteAtlasiin).

To evaluate if the same metrics would be found, another researcher coded metrics from two papers. Capture-recapture method[32] was then used which showed that 90% of metrics were found.

Table 2: Publication distribution of primary documents

Publication channel	Type	#	%
Agile Conference	Conference	7	33
ICSE	Conference	2	10
ICSE	Workshop	2	10
XP Conference	Conference	2	10
Agile Development Conference	Conference	1	5
APSEC	Conference	1	5
ASWEC	Conference	1	5
ECIS	Conference	1	5
ECSA	Conference	1	5
Elektronika ir Elektrotechnika	Journal	1	5
Empirical Software Engineering	Journal	1	5
EUROMICRO	Conference	1	5
HICCS	Conference	1	5
ICSP	Conference	1	5
ICSSP	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

2.4 Quality assessment

Quality assessment form adopted from [9] was used to evaluate the quality of each primary document. 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 = should be already excluded
- 1 = contains 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

2.4.1 Data synthesis

Data synthesis followed the steps recommended by Cruzes et al [6]. 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 to represent them.

3. RESULTS - WHY AND HOW ARE METRICS USED

This chapter presents the preliminary results from the systematic literature review. Table 3 shows the distribution of primary documents by publication channels. Table 3 lists the primary documents and context info.

High level reasons for using measurements are listed in Table 4.

3.1 Iteration planning

Many metrics center around iteration planning. Tasks for the next iteration are selected and prioritized with the help of metrics.

Table 3: Overview of primary studies

ID	Year	Research method	Agile method	Team size	Domain
[4]	2011	Singlecase	FDDScrumMix	179 in 13 teams	Navigation system for automobiles
[5]	2009	Multicase	NA/Scrum/Scrum	9/7/5	ERP/Graphic design plug-in/Facility management
[10]	2002	Single-experience	XP	50	Enterprise resource solution for the leasing industry
[25]	2010	Singlecase	ScrumXPMix	Dev site 600	Telecom
[27]	2011	Singlecase	ScrumXPMix	Dev site 500	Telecom
[26]	2010	Singlecase	AgileMix	6-7	Telecom
[33]	2011	Singlecase	ScrumBan	6-8	Telecom maintenance
[34]	2007	Singlecase	ScrumXPMix	4	Independent software developer
[38]	2006	Singlecase	XPMix	15	Enterprise information system
[36]	2010	Singlecase	LeanMix	project size 100	Telecom
[35]	2011	Singlecase	LeanMix	project size 200	Telecom
[29]	2010	Singlecase	LeanMix	NA	Telecom
[17]	2011	Singlecase	LeanScrumFDD	5±2	Information and communication software development
[18]	2012	Single-experience	XPMix	NA	Web application development
[30]	2011	Single-experience	AgileMix	9 and 6	Casino games
[37]	2009	Singlecase	XP	15	Enterprise information system
[39]	2006	Single-experience	ScrumXPMix	NA	Telecom
[11]	2011	Survey	Scrum	26 teams	Desktop and SaaS products
[12]	2010	Single-experience	Scrum	5-9	NA
[14]	2004	Multicase	XP/Scrum	4-18/6-9	b-2-b e-commerce solutions/Criminal justice system development
[23]	2012	Singlecase	Scrum	6-8	Web page development
[40]	2013	Singlecase	ScrumMix	5	Space mission control software
[7]	2013	Single-experience	Scrum	25 teams	Software for oil and gas industry
[28]	2012	Singlecase	LeanMix	NA	Telecom
[24]	2007	Singlecase	Lean	Comp. 160 devs	Various
[15]	2007	Singlecase	ScrumMix	500	Security services
[8]	2005	Singlecase	XP	15	Enterprise information system

Table 4: Categories for why and how measurement usage

Categories	Sources
Iteration planning	[10][30][5][12][16] [23][13][15][15]
Iteration tracking	[27][38][23][8][16] [7][11][10][24][39] [40][14][36][33][30] [12][29]
Motivate and enable team to improve	[39][37][17][7][5] [30][36][38]
Problem identification	[27][39][24][17][36] [23][26][34][25][41] [28]
Pre-release quality	[4][18][18][8][39]
Post-release quality	[26][5][11][36]
Changes in processes or tools	[17][27][25][7][36] [18][35][29]

Many metrics were focused to help in the prioritization of tasks for the next iteration [12][13][15]. Effort estimation metrics were used scope out features that would not fit to the iteration[10]. Consequently, velocity metrics are used to calculate how many features is the team able to complete in an iteration[30]. Velocity metrics can be also used to improve next iterations estimates[23]. Also, task start and end date metric was used to point out interdependent tasks in planning phase[16]. Knowing the teams' effective available hours is also useful when selecting tasks for a iteration[5]. Measuring the completion of tasks enables selecting incomplete tasks to the next iteration[16]. Prioritization of features can also be affected by a metric that measures the amount of revenue a customer is willing to pay for a feature[15]. Higher valued customers get their features done first.

3.2 Iteration tracking

Purpose of iteration tracking is to make sure that the tasks selected for the iteration are completed or that necessary modifications are done to the plan to end the iteration according to schedule. Metrics help in monitoring, identifying problems and predicting end result.

Progress metrics made it transparent to stakeholders how the iteration is progressing. [27][38][23] [8][16][11] [39][40]. Progress metrics included number of completed web pages[16], story completion percentage [39] and velocity metrics [8]. However, using velocity metrics can also have negative effects such as cutting corners in implementing features to maintain velocity with the cost of quality[10]. One unorthodox progress metric was product demonstrations with customer [40].

If it seemed that not all planned tasks could be completed, tasks were cut from the iteration [23][8][24] or extra resources were added [8][24].

When there were problems that needed to be fixed - metrics helped in making decisions to fix them - whether they were short or long term [36][8][27][7]. It was possible to base decisions on data, not only use common sense and experience [37]. Balance of work flow was mentioned as a goal for using metrics in multiple papers [30][26][29][12][27][8][17]. Crosstraining people to work on multiple disciplines was used to balance the work flow[24][37]. Also, metrics were used focus work on tasks that matter the most, avoid partially done work, task switching[33] and polishing of features[37]. Typically an iteration ends in a release but if too many defects are found the release must be delayed as happened in one case [14].

3.3 Motivate and enable team to improve

This section describes metrics that are used to motivate people and make them improve on their own.

Metrics were used to communicate different data about the project or product to the team members [39][37][30][36][38]. Measurement data motivated teams to act and improve their performance[37][30][5][7][17]. Some examples include fixing the build faster[17][7], fix bugs faster[5] or creating unit tests to declare the completion of a feature[37].

Metrics can be also used to prevent harmful behaviour such as cherry picking tasks that are most interesting. Measuring work in progress and setting WIP limits prevent cherry picking by enforcing only few work items at a time.[24]

3.4 Problem identification

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

There were multiple cases highlighting how measures are used to identify or predict problems so that the problems could be solved or avoided [27][39][23][26][34][25][41].

Sometimes there can be work phases where no value is added for example "waiting for finalization". This type of activity is called waste and can be identified by using lead time. [28]

Creating awareness with defect trend indicator helped to take actions to avoid problems [36]. One observed solution to problems was to find the root cause[17][24].

3.5 Pre-release quality

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

Integration fails were a common problem to avoid with code metrics[4][18]. Moreover, metrics were used to make sure that the product is sufficiently tested before the next step in the release path[18][8]. Additionally, making sure that the product is ready for further development was mentioned. Also, using metrics to improve pre-release quality was goal in one case[18]. Some metrics force the approach of writing tests before actual code[39]. Technical debt was measured with a technical debt board which was used to facilitate discussion on technical debt issues [7].

3.6 Post-release quality

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

Customer satisfaction, customer responsiveness and quality indicators are seen as attributes of post-release quality. Some metrics include customer input to determine post-release quality[26] [11][5][] while other metrics use pre-release data as predictors of external quality[36] [26][11]. Customer related metrics include for example defects sent by customers[5], change requests from customers[26] and customer's willingness to recommend product to other potential customers[11]. Quality prediction metrics include defect counts[26], maintenance effort[36] and deferred defect counts[11].

3.7 Changes in processes or tools

This chapter lists what kind of changes metrics had for processes and tools.

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[17].

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.[29]

Changes to requirements managements were also done based on metrics in an other case at Ericsson[25].

The problem with broken build, and the long times to fix the build, gave birth to measurements that monitor and visualize the state of the build and the time to fix it[7][17][18].

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 the release[17][18]. Similarly, testing approaches were changed based on flow metrics[25][35].

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 through demonstrating the product to the customer [40].

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 help keeping the internal quality of the product high throughout the development which then provides safe development of modifications or 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 are 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 [7] 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. From Eetu ▶*Tahan saattaisi loytya kaivamalla lisaa tarvittaessa.*◀

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 [16] were the primary progress measure. However, some other measures from Section section 3.2 show that instead of working code agile teams followed completed tasks and velocity metrics. From Mika ▶*mita tarkoittaa risk management interation tracking kohdassa en tajua*◀ From Eetu ▶*Kommentoin risk managementin pois. Se oli osittain paallekkaista kamaa Problem Identificationin kanssa. + melkein kaikki kama israelin casesta*◀

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 Mika ▶*paljon vittauksia Iteration tracking kappaleeseen. Onkohan kaikki oikein?*◀ From Eetu ▶*Naytti olevan ihan oikein. IterationTracking verkko taitaa sisaltaa eniten tavaraa.*◀

4.2 Implications for research

It was interesting to notice that there wasn't many code metrics, only the ones mentioned in [4][18] 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. I have 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
- Motivate and enable team to improve
- Problem identification
- 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 kontekseteja 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] P. Brereton, B. A. Kitchenham, D. Budgen, M. Turner, and M. Khalil. Lessons from applying the systematic literature review process within the software engineering domain. *Journal of systems and software*, 80(4):571–583, 2007.

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

[4] M. Cataldo and J. Herbsleb. Factors leading to integration failures in global feature-oriented development: An empirical analysis. In *Proceedings - International Conference on Software Engineering*, pages 161–170, Waikiki, Honolulu, HI, 2011.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] T. Dybå and T. Dingsøyr. Empirical studies of agile software development: A systematic review. *Information and Software Technology*, 50(9Ű10):833–859, Aug. 2008.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

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

[20] B. Kitchenham. WhatŠs up with software metrics? Ű a preliminary mapping study. *Journal of Systems and Software*, 83(1):37–51, Jan. 2010.

[21] 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.

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

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

[24] 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.

[25] 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.

[26] 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.

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

[28] 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.

[29] 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.

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

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

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

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

[38] 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.

[39] V. Trapa and S. Rao. T3 - tool for monitoring agile development. In *Proceedings - AGILE Conference, 2006*, volume 2006, pages 243–248, Minneapolis, MN, 2006.

[40] 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.

[41] 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.

[42] J. Webster and R. T. Watson. Analyzing the past to prepare for the future: Writing a. *MIS quarterly*, 26(2), 2002.

APPENDIX

A. BLAA