Product Evaluation Through Contractor and In-House Metrics

LUCIJA BREZOČNIK AND ČRTOMIR MAJER, University of Maribor

Agile methods are gaining in popularity and have already become mainstream in software development due to their ability to produce new functionalities faster, and with higher customer satisfaction. Agile methods require different measurement practices compared to traditional ones. Effort estimation, progress monitoring, improving performance and quality are becoming important as valuable advice for project management. The project team is forced to make objective measures to minimise costs and risks with rising quality at the same time. In this paper, we merge two aspects of agile method evaluation (the contractor and the client view), propose AIM acronym and discuss two important concepts in order to perform objective measurements: "Agile Contractor Evaluation" (ACE) and "Agile In-House Metrics" (AIM). We examine what type of measurements should be conducted during agile software development methods and why.

Categories and Subject Descriptors: **D.2.8 [Software Engineering]:** Metrics—Performance measures; Process metrics; Product metrics

General Terms: agile software development, agile metrics, agile contractor evaluation

Additional Key Words and Phrases: agile estimation

1. INTRODUCTION

The transition from the waterfall development process and its variations to an agile one poses a challenge for many companies [Green 2015, Laanti et al. 2011, Schatz and Abdelshafi 2005, Lawrence and Yslas 2006]. Examples of organisations that have successfully carried out the transition are Cisco [Cisco 2011], Adobe [Green 2015], Nokia [Laanti et al. 2011], Microsoft [Denning 2015], and IBM [IBM 2012]. However, not all companies have the same aspirations with regard to the reason they want to introduce an agile approach [VersionOne 2016]. The most common reasons include: to accelerate product delivery, to enhance the ability to manage changing priorities, to increase productivity, to enhance software quality, etc. However, the metrics of success need to be selected wisely. Based on the recently released 10th Annual State of Agile Report [VersionOne 2016], the main metrics are presented in Figure 1.

A majority of agile methods share an important aspect in terms of development planning. Each determines the preparation of prioritised features that need to be done (e.g. Product Backlog in Scrum). Developers pull features from the list in relation to the capacity that is currently available, i.e. if a company uses Scrum, the pull of features is used only at the beginning of each iteration (sprint). In the case of Kanban, the pull of features is continuous. Because agile methods are about team and teamwork, they all determine some kind of regular interaction between the development team and management, and communication within the development team.

The agile metrics are widespread in agile companies in order to monitor work and tend to make improvements inside them. In this paper, we will discuss the metrics that are used in agile software development from the point of "Agile Contractor Evaluation" (ACE) and "Agile In-House Metrics" (AIM). ACE covers the approaches for monitoring the progress of agile contractors, while AIM

This work is supported by the Widget Corporation Grant #312-001.
Author's address: L. Brezočnik, Faculty of Electrical Engineering and Computer Science, University of Maribor, Smetanova 17, 2000 Maribor, Slovenia; email: lucija.brezocnik@um.si; Č. Majer, Faculty of Electrical Engineering and Computer Science, University of Maribor, Smetanova 17, 2000 Maribor, Slovenia; email: crtomir.majer1@um.si.

In: Z. Budimac, Z. Horváth, T. Kozsik (eds.): Proceedings of the SQAMIA 2016: 5th Workshop of Software Quality, Analysis, Monitoring, Improvement, and Applications, Budapest, Hungary, 29.-31.08.2016. Also published online by CEUR Workshop Proceedings (CEUR-WS.org, ISSN 1613-0073)

approaches are used for monitoring work and workflow within the company. All metrics discussed in this paper were obtained on the basis of preliminary research.

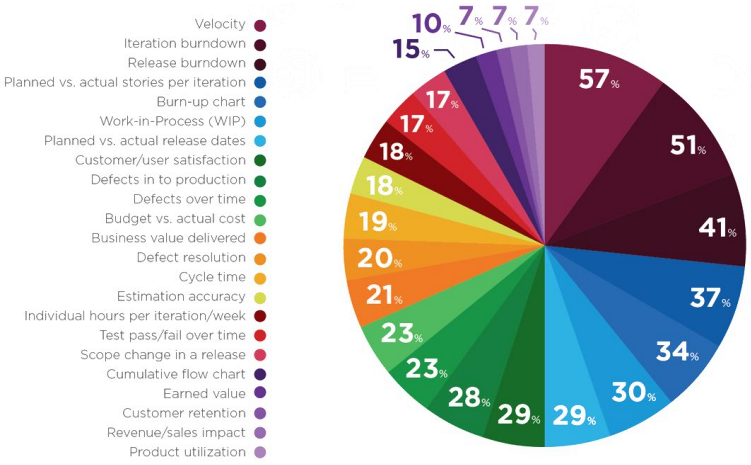

Fig. 1. Success measured on a Day-by-Day basis

During this paper, we will address the following two research questions:

- RQ1: What are the metrics and their usage in agile project management within the company?
- RQ2: How to evaluate an agile contractor?

2. RELATED WORK

The topic of software metrics has been covered well in several scientific papers. In this section, we mention only a few that are directly relevant to our research.

In the paper "Model of Agile Software Measurement: A Case Study" [Gustafsson 2011] it is stated that metrics are obtained in the form of a research method case study. Some authors (e.g. [Drury-Grogan 2014]) have conducted an interview with the intention of identifying iteration objectives and critical decisions, which refer to the golden triangle. A commonly used research method is also surveyed, e.g. described in [Lei et al. 2015]. The main purpose of it is to identify the effect of agile metrics Scrum and Kanban on software development projects backed up by statistical analysis. Some research done focuses only on identifying metrics for the improvement of specific aspects of the business, e.g. business value [Hartmann and Dymond 2006]. In [Qumer and Henderson-Sellers 2008] the authors have proposed a framework to support the evaluation, adoption and improvement of agile methods in practice. Some authors have tackled the problem of metrics in a comprehensive way and carried out a systematic literature review [Kupiainen et al. 2015]. However, in none of the analysed articles were the AIM and ACE metrics collectively accounted. This is why we committed ourselves to this article.

3. AGILE IN-HOUSE METRICS (AIM)

With the term "Agile In-house Metrics" we are referring to metrics that are measured in software development companies. With them, we are trying to predict delivery times, costs and delivery capacity, so that all initially planned features will be realised. We present the most commonly recommended metrics and allocate them in the following four main sections [Gustafsson 2011, Kammelar 2015]: lean metrics, business value metrics, quality metrics, and predictability and team productivity metrics.

3.1 Lean metrics

The Lean software development is one of the most popular agile development methods [Poppendieck 2015]. It promotes seven software development principles, which are very similar to those in lean manufacturing, initially used by Toyota [Gustafsson 2011]. Lean focuses on optimising flow efficiency across the entire value stream [Poppendieck 2015]. Lean principles are [Poppendieck 2015]: eliminate waste, amplify Learning, decide as Late as Possible, deliver as Fast as Possible, empower the Team, build Quality in, and see the Whole.

The most commonly used lean metrics are the following: *Lead time* is measured by elapsed time, such as minutes, or hours for one feature to go through the entire process from start to finish [Mujtaba et al. 2010]. *Optimal Lead time* is as short and stable as possible. It can be used to schedule high priority features and plan demo dates with customers [Kupiainen et al. 2015, Polk 2011] or for monitoring work in progress to predict a project schedule. On the other hand, we need to be careful if the Lead time is used as a metric for productivity. Bug fixes and quality improvement demand time.

Work in Progress (WIP) represents the set of features development team is working on [Middleton et al. 2007]. For controlling the mentioned Lead time, WIP constraints are used.

The *Queue* is another metric and is defined in [Staron and Meding 2011] as the "number of units remaining to be developed/processed by a given phase or activity." Unmanaged queues have an effect on Lead time and workflow such as increased Lead time and ineffective flow, respectively.

Cycle Time is measured by the amount of time per unit, where the unit means, for example, hours/feature. In other words, it is the amount of time it takes for a specific feature to be completed [Polk 2011].

3.2 Business Value Metrics

The business value metrics are intended for long-term business process monitoring. Every business tries to get a high return on investment (ROI) and profit for any investment or at least to break even. It is important to highlight that ROI may not always be measured in the financial aspect. *Earned Business Value* (EBV) [Javdani 2013] metrics may be used in conjunction with the ROI, where estimated ROI is prorated to the features of each iteration in the financial aspect.

One of the commonly recommended metrics for business value is also the *Customer Satisfaction Survey*, where we ask the customers if they are satisfied with the product and *Business Value Delivered*. The latter can be interpreted by delivered features per timeframe [Kupiainen et al. 2015, Abbas 2010]. To estimate future expenses in software development, the *Average Cost per Functions* (ACPF) [Gustafsson 2011] metric is recommended.

3.3 Quality Metrics

As presented in the paper "Metrics and Models in Software Quality Engineering" [Kan 2002], the quality metrics are classified as Product metrics, Process metrics, or Project metrics. The product metrics monitor the characteristics of the product, e.g. size, complexity, and quality. The process metrics cover defect handling and can be used for software development improvement and maintenance. The project characteristic, e.g. cost, productivity, and schedule, are measured by project metrics. The following is a presentation of the most proposed metrics for quality.

The *Technical Debt* is a metric that is recommended by the Agile community [Behrens 2009]. It measures the amount of hours that are required to fix the software related issues [Kupiainen et al. 2015]. In the paper "Model of Agile Software Measurement: A Case Study" [Gustafsson 2011] says that is optimal to translate Technical Debt to money. In such an approach, it is hard to be accurate.

The *Defect Count*, another quality metric, counts defects in each iteration. It is important to highlight that defects can be viewed from different perspectives. One way is to use *Faults-Slip-Through* (FST) metrics [Damm et al. 2006] as a measurement of a number of defects that have not been found when it was most cost-effective. FST could also be used as a predictor of fault-proneness [Afzal 2010].

3.4 Predictability and team productivity metrics

Agile methods emphasise the predictable performance of the monitored process [Hayes et al. 2014]. Commonly recommended metrics in this section are as follows.

The *Velocity* can be measured directly or indirectly [Padmini et al. 2015]. The direct approach is done using the monitoring volume of the work done by a given team in a given timeframe. It is important to stress that velocity is not measured by the ratio between the completed and committed stories (i.e. stories that were committed by developers to be completed in the iteration). If the development team realises that their work is primarily measured with it, they will use the following approach: "Tell me how you measure me, and I will tell you how I will behave" [Goldratt 1990]. Another negative part of the improper measurement of velocity as a productivity measure is the degradation of the quality of the developed software [Gustafsson 2011]. An indirect way of velocity measurement is done by bug correction time.

Running Automated Tests is the metrics for measuring a team's productivity by the size of the product [Gustafsson 2011]. It comprises all the features that are covered by automated tests.

4. AGILE CONTRACTOR EVALUATION (ACE)

In comparison to AIM, where measures are taken and evaluated inside a software development company, ACE describes ways to monitor the progress of agile contractors. The US Department of Defense [Hayes 2014] proposes the following list of activities for successful contractor evaluation during the project.

4.1 Software Size

For monitoring progress in traditional software development, the *size of the product* is often an acceptable indicator for eventual *costs* and *schedule*. Agile methods tend to fix costs and schedule using timebox planning or work in progress limits, leaving software size as the only variable [Lei et al. 2015]. The goal is to maximise the delivery of new functionalities to the customer, depending on work capacity via the story prioritisation (high priority stories first). In the agile world, software size is typically represented in story points or delivered user stories. Each iteration is planned with a subset of the most important user stories, that are planned to be completed according to team capacity. Therefore, minimising the difference between the number of story points planned and the number of stories delivered becomes the main focus. Size is used to help the team understand its capacity and serves as a basis for performance diagnosing. If a team has continuing problems with delivered story points, that indicates the capability of the team is not realistic and/or that the scope of work is not well understood [Hayes 2014].

4.2 Effort and Staffing

It is important to track *effort* and *staffing* since they tend to be the primary cost drivers. Using agile methods, there is a notable change in the staff utilisation pattern. During the early phases of development, a risk of slow ramp-up in staffing (the time until the team is fully formed) can appear. There also seems to be a decline in the number of participants towards the end of the development cycle, leading to possible complications [Hayes 2014].

4.3 Schedule

Tracking velocity over iterations and releases provide the basis for understanding the pace of delivery. It also helps with estimating the *completion time* of the remaining work in a backlog. With schedule analysis, the client can measure confidence in an agile contractor, regarding the timely completion of the agreed scope of work, and is thus able to identify potential risks early in the process. Confidence is a result of trust and previous experience with the contractor. A precise schedule is not a priority in

agile development, therefore focusing on things like schedule performance is less meaningful, in contrast to traditional approaches [Hayes 2014].

4.4 Quality and Customer Satisfaction

Agile methods put a lot of attention on the product quality, right from the beginning of a project. The acceptance criteria for each user story should be precisely defined, so that every product increment passes through multiple stages in the quality assurance process (unit testing, integration testing, UI testing, acceptance testing etc.), mitigating the risk of defects in a production environment. A customer plays a prominent role of prioritising user stories as well as providing timely feedback at the completion of each sprint. Defects, discovered and fixed during the quality assurance process inside each iteration/story provide the basis for quality metrics. This problem can also be addressed with the usage of Software Defect prediction techniques like presented in the paper "A Systematic Data Collection Procedure for Software Defect Prediction" [Mauša 2015].

The cumulative flow diagram is a powerful visualisation tool for early quality problem recognition. Figure 2 shows an example of such a diagram with an explanation of how to read different measures. It also displays two unhealthy states of a project due to inability to complete a development task, because of defects or other issues (A) and the poor rate at which new work is taken into the development process (B).

Patterns similar to (A) can lead to Technical Debt, which threatens the successful realisation of a project. Ideally, we want consistent and narrow bands for all measures, like the ones for validated and assigned categories. Pattern B shows that the development team is sitting idle or working on unplanned tasks out of iteration scope. Poor performance during one stage of the development process affects all the subsequent stages, which can be seen as leading to emerging gaps in the cumulative flow diagram [Hayes 2014].

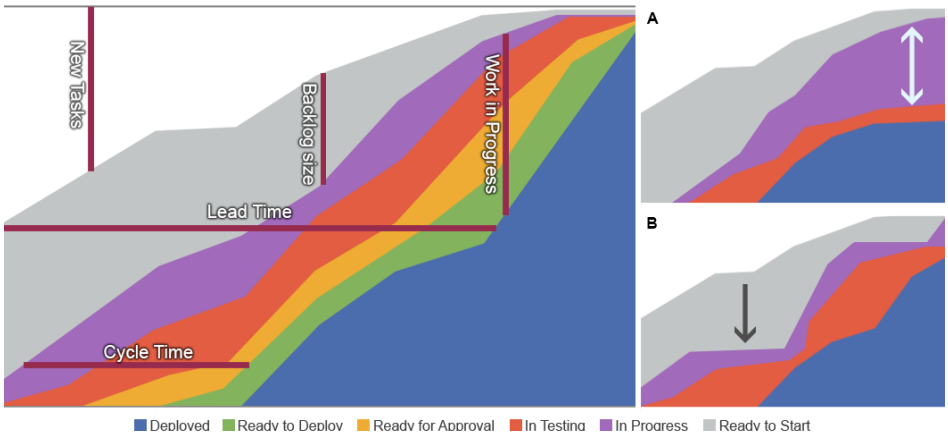

Fig. 2. How to read Cumulative Flow Diagram

4.5 Requirements

Requirements, typically represented as user stories, often change over time. Agile methods are very flexible about changes. In some environments, requirements in the backlog are less detailed, thus not providing all the necessary information for implementation. That information is acquired via a direct conversation with users or their surrogates which prolong the time of activity completion. In other environments, user stories are fully defined, including information about "done criteria" and test cases used for verifying the product increments. Metrics that reveal how well-defined are the requirements, include: a number of changes in user stories, the number of items in a backlog (where "done criteria"

needs to be specified), the cycle time between a customer identifying the new story as a top priority, and similar measures [Hayes 2014].

4.6 Delivery and Progress

Agile methods give product delivery a higher priority compared to other attributes such as costs, durations, and schedule. Frequent delivery of working software renders a more direct view of progress, allowing early opportunities to refine the final product and assure that the development team is moving toward the desired goal [Hayes 2014].

5. AGILE ESTIMATION

We presented agile methods and metrics used for project management. As can be seen, every project needs some estimation – costs, time or quality. Since low costs are not the first priority of the agile movement [VersionOne 2016], it leaves us with two variables. To estimate timeframe, we need to know the amount of work in the product/iteration backlog, which is defined with requirements and quality is best monitored through cumulative flow diagram.

For a typical Agile project, once a list of stories is made for iteration, the typical next steps are prioritisation and estimation. The overtime goal is to steady the number of story points delivered per iteration (work capacity), thus providing some fixed value for long-term scheduling [Zang 2008]. Story points per iteration, day, hour (any fixed unit of time) are often referenced as a metric named Takt Time. It is the rate at which product increments must happen in order to meet customer demand. Product Delivery Time, in means of time-limited increment, is then calculated as *Takt Time* × *Number of Story Points*. With a known number of story points and product deadlines, we can then calculate the necessary Takt time, to finish the product increment on time [Bakardzhiev 2014].

Forecasting software delivery dates is possible with different types of simulation such as the Monte Carlo Simulation. Monte Carlo Simulation is a problem-solving technique used to approximate the probability of certain outcomes by running multiple simulations using random variables. In our case, it provides for the probabilities of finishing all project activities at a given future date. In Figure 3, we can see the probability distribution of project completion, measured in the number of hours [Odenwelder 2015].

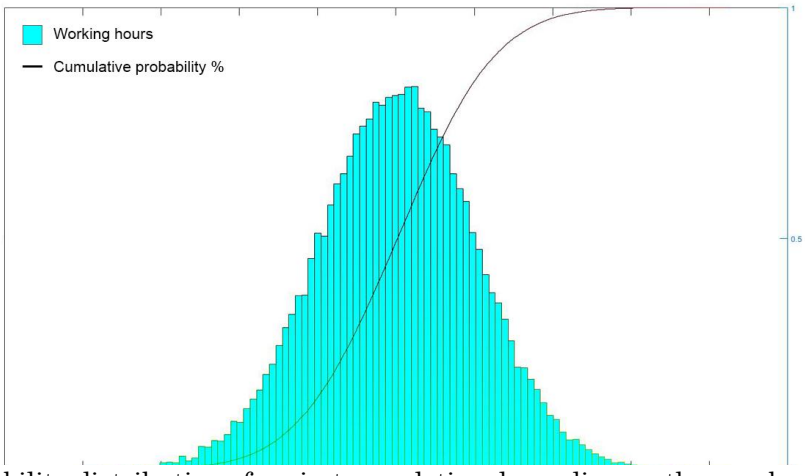

Fig. 3. Probability distribution of project completion depending on the number of hours

The simulation gives us a result – with a given Takt Time our team is able to deliver a completed product in around 500 work hours with a probability of 50%. However, this estimation is pretty risky.

A better estimation would be one where probability approaches 100% (that is around 530 work hours). The date of project completion is not a single value, but a set of possible results and it depends on us how much risk we are willing to take. There are multiple variations of Monte Carlo Simulation for agile estimation. We presented the basic one, where Takt Time is related to work hours, not taking task estimation errors or the stability of velocity into account. Regardless, from a simulation like this, we can get a better insight into how long a project can last, how many team members we need, what are the potential costs, what is the desired Takt Time, etc.

6. DISCUSSION

Our goal to propose the acronym AIM and endorse it by scientific papers is fully met. With it, we wanted to highlight the important influence of groups of metrics AIM for process improvement and ACE for the indirect assessment of the expected product. With this noted, we can answer the asked questions.

The RQ1 was aimed at discovering the metrics and their usage in agile project management within the company. We presented the most used and popular metrics, with which we could control the process of agile software development. The study showed that agile teams usually decide for velocity, burndown charts, planned vs actual, or work in progress. These metrics encourage the main objectives of agile – on time delivery, the ability to make changes quickly, a motivated team and, of course, a satisfied customer. Furthermore, metrics help us with decision making, optimising workflows, eliminating waste and making improvements over time. Our overview of metrics and measures can be used in a software development firm, to increase awareness among management, agile practitioners, support and within the team itself, to assist the company's growth.

The second question, RQ2, was aimed at finding how to evaluate a contractor. While most papers try to answer RQ1, we also gave some attention to the customer side – how do they evaluate an agile contractor? Customers are typically not as well educated about agile practices. Therefore, they might not know what to monitor, measure, and evaluate. We presented key perspectives and practices for the customer side, to understand how well a contractor performs on a daily basis and overall. With the help of these practices, a customer can quickly notice difficulties in the project and warns the contractor about them – this way, both parties participate in reducing risk and improving their trust.

Our future plans are to test proposed metrics in a real life scenario in the form of a research method Case study or Experiment. We would also like to verify whether and to what extent the Monte Carlo method assists in agile estimation and conduct additional research on the evaluation of products.

REFERENCES

N. Abbas, A.M. Gravell, and G.B. Wills. 2010. The impact of organization, project and governance variables on software quality and project success. In *Proceedings of Agile Conference 2010 (AGILE 2010)*, 77–86.

Wasif Afzal. 2010. Using Faults-Slip-Through Metric As A Predictor of Fault-Proneness. In *Proceedings of Asia Pacific Software Engineering Conference 2010*, 414–422. DOI:http://dx.doi.org/10.1109/APSEC.2010.54

Dimitar Bakardzhiev. 2014. #NoEstimates Project Planning Using Monte Carlo Simulation. Retrieved June 3, 2016, from https://www.infoq.com/articles/noestimates-monte-carlo

Pete Behrens. 2009. Scrum Metrics and Myths Available. Retrieved June 3, 2016, from http://scrumorlando09.pbworks.com/w/page/15490672/ScrumMetrics-and-Myths

Cisco. 2011. Agile Product Development at Cisco: Collaborative, Customer-Centered Software Development. Retrieved June 3, 2016, from http://goo.gl/OS2gmH

Lars-Ola Damm, Lars Lundberg, and Claes Wohlin. 2006. Faults-slip-through—a concept for measuring the efficiency of the test process. *Software Process: Improvement and Practice*, 11 (2006), 47–59.

Steve Denning. 2015. Surprise: Microsoft Is Agile. Retrieved June 3, 2016, from http://www.forbes.com/sites/stevedenning/2015/10/27/surprise-microsoft-is-agile/#2eeb35714b9e

Meghann L. Drury-Grogan. 2014. Performance on agile teams: Relating iteration objectives and critical decisions to project management success factors. *Information and Software Technology* 56, 5 (2014), 506–515. DOI:http://dx.doi.org/10.1016/j.infsof.2013.11.003

Eliyahu M. Goldratt. 1990. The Haystack Syndrome: Sifting Information Out of the Data Ocean.

Peter Green. 2015. Early Agile Scaling at Adobe. Retrieved June 3, 2016, from http://agileforall.com/early-agile-scaling-at-adobe/

Johan Gustafsson. 2011. Model of Agile Software Measurement: A Case Study. Master of Science Thesis in the Programme *Software engineering and Technology*. Retrieved June 1, 2016, from http://publications.lib.chalmers.se/records/fulltext/143815.pdf

Deborah Hartmann and Robin Dymond. 2006. Appropriate agile measurement: using metrics and diagnostics to deliver business value. In *Proceedings of Agile Conference (AGILE'06)*. Minneapolis, pp. 6 pp.–134. DOI:http://dx.doi.org/10.1109/AGILE.2006.17

Will Hayes, Suzanne Miller, Mary Ann Lapham, Eileen Wrubel, and Timothy Chick. 2014. *Agile Metrics: Progress Monitoring of Agile Contractors*. Technical Note CMU/SEI-2013-TN-029. Carnegie Mellon University, Pittsburgh, PA.

IBM. 2012. How IBM saved $300 million by going agile. Retrieved June 3, 2016, from https://www.ibm.com/developerworks/community/blogs/invisiblethread/entry/ibm-agile-transformation-how-ibm-saved-300-million-by-going-agile?lang=en

Taghi Javdani, Hazura Zulzalil, Abdul Azim Abd Ghani, and Abu Bakar Md Sultan. 2013. On the Current Measurement Practices in Agile Software Development. *International Journal of Computer Science Issues* 9, 3 (2013), 127–133.

K. V. Jeeva Padmini, H. M. N. Dilum Bandara, and Indika Perera. 2015. Use of Software Metrics in Agile Software Development Process. In *Proceeding of the Moratuwa Engineering Research Conference (MERCon)*. Moratuwa, 312–317. DOI:http://dx.doi.org/10.1109/MERCon.2015.7112365

John Kammelar. 2015. Agile Metrics What is the state-of-the-art?. Retrieved June 2, 2016, from http://www.metrieken.nl/wp-content/uploads/2013/11/Agile-Metrics-by-John-Kammeler-1.0.pdf

Stephen H. Kan. 2002. Metrics and Models in Software Quality Engineering.

Eetu Kupiainen, Mika V. Mäntylä, and Juha Itkonen. 2015. Using metrics in Agile and Lean Software Development – A systematic literature review of industrial studies. *Information and Software Technology* 62 (2015), 143–163. DOI:http://dx.doi.org/10.1016/j.infsof.2015.02.005

Maarit Laanti, Outi Salo, and Pekka Abrahamsson. 2011. Agile methods rapidly replacing traditional methods at Nokia: A survey of opinions on agile transformation. *Information and Software Technology* 53, 3 (2011), 276–290. DOI:http://dx.doi.org/10.1016/j.infsof.2010.11.010

Richard Lawrence and Blas Yslas. 2006. Three-way cultural change: Introducing agile with two non-agile companies and a non-agile methodology. In *Proceedings of AGILE Conference 2006 (AGILE'06)*, 255–262. DOI:http://dx.doi.org/10.1109/AGILE.2006.57

Howard Lei, Farnaz Ganjeizadeh, Pradeep Kumar Jayachandran, and Pinar Ozcan. 2015. A statistical analysis of the effects of Scrum and Kanban on software development projects. *Robotics and Computer-Integrated Manufacturing* (2015). DOI:http://dx.doi.org/10.1016/j.rcim.2015.12.001

Goran Mauša, Tihana Galinac Grbac, and Bojana Dalbelo Bašić. 2015. A Systematic Data Collection Procedure for Software Defect Prediction. *Computer Science and Information Systems* 13, 1 (2015), 173–197. DOI: 10.2298/CSIS141228061M

P. Middleton, P.S. Taylor, A. Flaxel, and A. Cookson. 2007. Lean principles and techniques for improving the quality and productivity of software development projects: a case study. *Int. J. Prod. Qual. Manage.* 2, 4 (2007), 387–403.

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

Jerry Odenwelder. 2015. How to Reduce Agile Risk with Monte Carlo Simulation. Retrieved June 3, 2016, from https://blogs.versionone.com/agile_management/2015/03/21/how-to-reduce-agile-risk-with-monte-carlo-simulation/

R. Polk. 2011. Agile & kanban in coordination. In *Proceedings of Agile Conference 2011 (AGILE'11)*, 263–268.

Mary Poppendieck. 2015. Lean Software Development: The Backstory. Retrieved June 3, 2016, from http://www.leanessays.com/2015/06/lean-software-development-history.html

A. Qumer and B. Henderson-Sellers. 2008. A framework to support the evaluation, adoption and improvement of agile methods in practice. *Journal of Systems and Software* 81, 11 (2008), 1899–1919. DOI:http://dx.doi.org/10.1016/j.jss.2007.12.806

B. Schatz and I. Abdelshafi. 2005. Primavera Gets Agile: A Succesful Transition to Agile Development. *IEEE Software* 22, 3 (2005), 36–42. DOI:http://dx.doi.org/10.1109/MS.2005.74

M. Staron and W. Meding. 2011. 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.

VersionOne. 2016. 10th Annual State of Agile Report. Retrieved June 3, 2016, from http://stateofagile.versionone.com/

Juanjuan Zang. 2008. Agile Estimation with Monte Carlo Simulation. In Proceedings of 9th International Conference, XP 2008. Ireland, 172–179. DOI:http://dx.doi.org/10.1007/978-3-540-68255-4_17