The Journal of Systems and Software 117 (2016) 258–273

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Quantitatively measuring a large-scale agile transformation

Marta Olszewska (née Pląska)[a,b,*], Jeanette Heidenberg[c], Max Weijola[a,b], Kirsi Mikkonen[c], Ivan Porres[a,b]

[a] Åbo Akademi University, Faculty of Natural Sciences and Engineering, Informationsteknologi, Domkyrkotorget 3, Åbo 20500, Finland
[b] Turku Centre for Computer Science, Domkyrkotorget 3, Åbo 20500, Finland
[c] Ericsson R&D Center Finland, Hirsalantie 11, Jorvas 02420, Finland

ARTICLE INFO

Article history:
Received 3 June 2015
Revised 10 February 2016
Accepted 8 March 2016
Available online 21 March 2016

Keywords:
Metrics
Transformation
Agile

ABSTRACT

Context: Agile software development continues to grow in popularity and is being adopted by more and more organizations. However, there is a need for empirical evidence on the impact, benefits and drawbacks of an agile transformation in an organization since the cost for such a transformation in terms of money, disrupted working routines and quality of development can become considerable. Currently, such evidence exists in the form of success stores and case studies, mostly of qualitative nature.

Objective: Provide a metrics model to quantitatively measure the impact of an agile transformation in a software development organization.

Method: The metrics model was elicited with the use of the Goal Question Metric approach.

Results: A quantitative metrics model containing eight rigorously described metrics is presented and followed by its application to evaluate an agile and lean transformation in a large international telecommunication organization with 350 employees in two sites.

Conclusions: The metrics model was sensitive to the changes that occurred in the organization and revealed significant improvements in six of the eight metrics and a deterioration in one of the metrics.

1. Introduction

The IT world of today is highly competitive and value oriented. Companies active in this field strive to be flexible and adaptive to change due to constant business and technological changes in requirements and in the environment. As a consequence, agile and lean software development methods are gaining popularity among companies of various sizes and domains (Rodríguez et al., 2012; Dingsøyr and Moe, 2013). Agility is in itself a desired characteristic with over a decade of successful adoption both as a new development process or a changes in existing processes.

Transforming the way of working in a large organization takes time, effort and resources and there is a need to constantly evaluate the need and benefits of such initiatives. Software metrics can be used to provide objective insights and evaluate the effect of software process changes (Kitchenham, 1996).

One challenge when trying to apply process metrics to evaluate a transformation is that metrics can be process specific. That is, a metric is related to a task, role or artifact that is defined in one way of working or process but not in the other. We try to overcome this and other challenges related to application of software metrics in the study of large agile transformations and formulate the research question to be answered in this study as:

RQ: *How can the changes of an agile transformation be measured by quantitative objective metrics?*

To answer this question, we present a metrics model to measure the impact of agile transformations and its application in the context of a large-scale telecommunication company. In particular, we use metrics and measurements that are feasible to analyze both plan driven development and agile and lean development. The goal is to use metrics and measurements for the purpose of comparing the state of the organization before and after an agile and lean transformation. However, the metrics can be applied with other purposes, for instance, to provide transparency, feedback and aid for the self-organization of teams.

* Corresponding author at: Åbo Akademi University, Faculty of Natural Sciences and Engineering, Informationsteknologi, Domkyrkotorget 3, 20500 Åbo, Finland. Tel.: +358 442806858.
E-mail addresses: mplaska@abo.fi (M. Olszewska (née Pląska)), kirsi.mikkonen@ericsson.com (K. Mikkonen).

http://dx.doi.org/10.1016/j.jss.2016.03.029
0164-1212/

We use as the starting point the metrics model presented in our previous work, where we proposed a set of metrics for evaluation of improvement of the development process. In this paper we refine the earlier established metrics model and the metrics themselves. We base our enhancement on iterative collaboration with practitioners, e.g. via workshops and meetings. Thus, we develop the model based on empirical data. We formulate the goal of measurement in a neutral manner, so that the change itself is assessed—not the improvement. We depict the extended metrics in detail by using structured and formalized description.

Our contribution in this paper is as follows. We provide a structure for the description of metrics and thoroughly define each metric accordingly. Moreover, we discuss the validity and appropriateness of the proposed metrics for their intended use. Furthermore, we apply the metrics in a study of a large organization from the telecommunication domain. We perform validation on two levels: (i) when presenting the analysis and investigating the weaknesses and strengths of the proposed metrics, (ii) when discussing the validity of the study. Finally, we examine our results on a generic level, i.e., the feasibility of the proposed metrics for the purpose of the study.

Our results are presented in such a way that organizations of similar size and organizational context could use as a reference to monitor an agile and lean transformation. Experience and continuous learning are central tenets in both the agile and lean communities. We hope that our results contribute to this idea by extending the body of knowledge in the agile and lean domain as well as enabling further empirical investigations by providing a re-usable metrics model.

This paper is structured as follows. Section 2 introduces some relevant terminology and related work, followed by a description of the research question and the investigation strategy in Section 3. In Section 4 the metrics model is introduced at a high level, whereas each metric is described in detail in Section 5. Following the metrics description, the main results of an empirical study are presented in Section 6. Section 7 contains a discussion on validity concerns regarding the metrics. Our conclusions are presented in the last section.

2. Background and related work

There are plenty of existing metric models available, just to mention the ones of Boehm (1978), McCall et al. (1977) (known as General Electric Model) and Dromey (1995), or standards that include quality models, e.g. SQuaRE (ISO/IEC, 2010), ISO 9126 (ISO, 2001) and ISO 9004:2000 (ISO, 2000). However, tailoring measurements for specific context might require the “define-your-own-model” approach (Gilb, 1988), especially in order to encapsulate both product and process metrics. We use the “define-your-own-model” approach and support it with Goal Question Metric method (Basili et al., 1994).

2.1. Empirical investigations

The topic of empirically comparing different development processes, regardless if the results are of quantitative or qualitative nature, has been identified as relevant by several researchers. The qualitative evidence on how the adoption of software process practices can affect software product industry was described in Rönkkö et al. (2012), where authors based their findings on a survey study. The authors were investigating the use of agile methods with respect to the maturity level of companies included in the survey. We however, do not take into account maturity level of the case organization and focus on quantitative data.

Very few reports on quantitative, empirical studies comparing the situation before and after an agile transformation exist. In the work of Concas et al. (2012), software quality metrics were related with certain agile development phases and practices. Our work depicts higher level viewpoint, meaning that we study the changes in the development process as a whole, without focusing on particular development phases.

Qualitative study of a comparative manner, which was based on analysis of results of traditional software metrics, was done for five systems developed with agile and three with plan-driven methodologies (Destefanis et al., 2014). Authors focused on the comparison of the code on the class level using existing metrics, whereas we concentrated on developing metrics model and refining metrics which would assess the development on the level of organization and the change of its development process.

The impact and benefits of agile and lean software deployment have been investigated previously and reported in Šmite et al. (2010); Anderson (2004); Leffingwell (2007); Parnell-Klabo (2006); Petersen and Wohlin (2010). Most of these studies are, however, of a qualitative nature. As other authors have pointed out, there is a need for further empirical studies (Ebert et al., 2012; Dingsøyr and Moe, 2013), but also for quantitative results (Dybåand Dingsøyr, 2008).

2.2. Measuring transformation

The topic of comparison of various development processes, and in particular shifting form one to another, is still in need for further study (Sjøberg et al., 2012; Petersen and Wohlin, 2010). In particular in Petersen and Wohlin (2010) it is visible that there is a need for collecting empirical data to demonstrate the impact of changes in the development process. The authors use quantitative data to either confirm or contradict the information given by qualitative data. We believe that quantitative data can be used not only as a reporting mechanism and self improvement driver, but also a good verifier for the human perception that can be successfully juxtaposed with qualitative data.

One report that shares our focus on the plan-driven “old” versus agile “new” development methods is presented by Li et al. Li et al. report a longitudinal case study with focus on software quality, comparing a plan-driven development setting with Scrum. In contrast to the Li et al. study, in our work we aim to focus on a bigger set of attributes than purely software quality. A study that compares the effects of moving from a plan driven to an agile approach to software development is described by Petersen and Wohlin (2010). Authors focus on qualitative data gathered from interviews, with only a few quantitative performance metrics included for support. In our work, interviews and other communication means were used to discuss the metrics model and discovered metrics, as well to clarify some inconsistencies in raw data.

Agile adoption from defect management viewpoint is presented in Korhonen (2014). It is based on experiences from three organizations in question (two fully agile and one using a mix of waterfall and agile processes) and provides results based on qualitative data, supported with quantitative data about defect count. We concentrate solely on quantitative data and have broader scope in our measurement process, than just defect-related.

The impact of agile transformation from a traditional, plan-driven method, was evaluated in Korhonen (2013), where a longitudinal case study was described. The assessment was done in terms of how well the organizational change succeeded in achieving the goals set by the management before the transformation started. The study included two separate analyses, made at six monthly intervals, which measured what agile practices were really in use in the organization, and what effects, if any, were visible in terms of the aforementioned goals. The study shares some similarities with our work, like the type of transformation (from plan-driven to agile), location, size and type of the case

organization (Finnish large telecommunication company). However, our work spans over the a longer period of time 2008-2012, in contrast to the investigation of early stage transformation. Moreover, our results are based solely on quantitative data, whereas in Korhonen (2013) the quantitative data are representing defects, and all the others are of qualitative nature (based on the opinion surveys of the personnel directly involved in the transformation).

Two studies that we found useful for our work are Petersen's and Wohlin's work on flow Petersen and Wohlin (2011), in combination with Staron and Meding's work on bottlenecks Staron and Meding (2011). The main difference is in the purpose of the research. Their purpose was to continuously improve an agile way of working, by analyzing the current way of working in order to find improvement opportunities.

3. Research question, method and context

In this section we describe our research by first explaining the research question, followed by the description of the context of the study and the data collection process.

3.1. Research question and research approach

The main premise of this article is the need to establish a model to measure the impact of a transformation to agile and lean methods Dybåand Dingsøyr (2008); Sjøberg et al. (2012). According to this, our main research question, as mentioned in the introduction, is

RQ - *How can the changes in the development process introduced by an agile transformation be measured by quantitative objective metrics?*

To elaborate the research question, we use the *goal template* by Basili and Rombach (1988); Wohlin et al. (2012) to define the goal of our study:

provide a metrics model applicable to plan driven and agile and lean development processes **for the purpose of** comparison
for the purpose of comparison
with respect to performance and quality
from the point of view of the organization and researchers
in the context of a large-scale telecommunication organization.

The aspects of the development process that we are interested to measure are the timeliness factor, i.e. the time (delay) between initiation and execution of a process (end-to-end lead time), delivering the business value (or product, feature, functionality, service, etc.) and effectiveness of the software development process. All these issues were identified as essential during discussions with the partnering organization. Naturally, both the organization and the researchers are also interested in the quality aspect of the development process in both time-frames.

To answer this question, we create a metrics model to be used for measuring an agile transformation. This metrics model should fulfill a number of constraints in order to be applicable for our goal. We consider that is particularly important for our metrics and measurements not only to be meaningful, useful and applicable, but also to be objective and available with respect to existing data. Each metric should preserve conceptual relationship with the attribute, i.e. be intuitive and easy to understand, as well as practical, i.e. be feasible to deploy in a software development setting. The data collection process should be mostly automatic and objective, i.e. no subjective judgment should be required to obtain the data. Summarizing, in order to ensure that the metrics model is useful for comparing the situation before and after a transformation, we set up the following criteria for selecting the metrics of the model:

- **C1:** The metrics must be applicable to both plan driven and agile projects.
- **C2:** The metrics must support the agile principles (as described in the agile manifesto Agile Alliance).
- **C3:** The metrics must be possible to collect and use in projects of any scope, size and complexity.
- **C4:** The metrics must be objective, i.e. metrics collection should not require the judgment and interpretation of experts.

We used the *Goal Question Metric* GQM method Basili et al. (1994) as described in Section 4 in order to develop our quality model. The metrics model in this paper has been developed and refined iteratively, in five steps, in a series of workshops with both industry experts and researchers. Much valuable knowledge have also been obtained through a study of the literature, described in Section 4.3, with the goal of discovering metrics already proven useful and relevant in literature. The metrics identified in the literature were cross-checked against the questions in our quality model, as well as the selection criteria presented in Section 3.1. Finally, a pilot evaluation was performed to exemplify the data gathering and visualization of a subset of metrics at Ericsson R&D Center Finland, further described in the following section.

3.2. Research context and unit of analysis

The key drivers for this research were identified in the *Cloud Software Finland* (Cloud Software Programme, 2013) project, where one of the purposes was to support Finnish software organizations in transforming their operations with the aid of agile and lean methods through cooperation with academic partners. The characteristics of the participating organizations varied not only with respect to their size, development processes and cultures, but also their profile, ranging from software development to consultancy services.

This paper focuses on a *large-scale* agile and lean setting. What constitutes large-scale is discussed by Dingsøyr and Moe (2013), where they present a number of factors that have been suggested as definition for : (i) project costs, (ii) project duration, (iii) size of the software developed, (iv) persons involved, (v) number of sites, (vi) number of teams. Dingsøyr and Moe focus on the number of teams in development and define large-scale agile as consisting of more than two teams (Dingsøyr and Moe, 2013; Dingsøyr et al., 2014).

In this paper we present the results of an iterative collaboration with *Ericsson R&D Center Finland*—a large-scale software development telecommunication company that has completed an agile and lean transformation. The organization itself, as well as the lean and agile transformation of that organization, has already been described in Heikkilä et al. (2013), however the focus was on the qualitative study. Yet, a need for further experimentation and providing quantitative measurements to complement the Heikkilä et al. (2013) study was identified by the case company. It was additionally motivated by the gap in the state of current research (Dybåand Dingsøyr, 2008).

The studied projects within Ericsson R&D Center consists of around 350 people, distributed in research and development centers in Finland and Hungary, with the major part of the development taking place in Finland. The large telecommunication product is roughly 10 years old and highly complex, consisting of RoseRT, C++ and Java code (Mikkonen et al., 2012). The case company had worked in a standardization-driven development style with teams focusing on different components in development silos. In 2010 the big transformation toward agile development was performed, breaking up the old silo structure and forming cross-functional agile teams. Around the beginning of 2011, the agile and lean way of

working emerged as the organization continued to grow into the new agile mindset.

For the new way of working, the case organization adopted Scrum for product development and Kanban for product maintenance (Mikkonen et al., 2012). The organization decided to adopt these approaches without extensive tailoring in order to ensure consistency between teams. Scrum was adopted as defined by Deemer et al. (2012), including the rearrangement of the physical environment and the introduction of the product owner and scrum master roles. On the other hand, the introduction of Kanban required the customization of the status columns in the Kanban boards and the adjustment of the work in progress limits (Mikkonen et al., 2012). The number of agile teams changed according to needs during the transformation from 1 team in 2009—pilot study, through 20 teams in 2011. The total number of teams was at the time of study more than 20 and hence fulfilled the definition of large-scale according to Dingsøyr and Moe (2013); Dingsøyr et al. (2014). The cross-functional and self-organizing teams of 6–7 people were built to be able to perform end-to-end development. However in practice, they received the features or tasks that were most suitable with respect to their previous experience and competence. This was caused by the specificity of a large and complex 10-year-old product.

The case organization emphasizes that their transformation is a "journey" meaning that they have not adapted agile and lean approaches in one step and that the principles and practices of agile were fine-tuned to the needs of the organization. For instance, the monolithic structure of the organization had to be decomposed to enable agile development process. There was a group of product owners established, headed by chief product owner, in order to cope with the distributed nature of development. Furthermore, there was a function of product manager responsible for long term planning of product development (product vision) and other product manager serving as a proxy between development and management in the organization. There was a separate product manager who was responsible for the early phases of the development process and related planning. Finally, the organization was assisted by experts from the telecommunication domain, to support the product management and development with their knowledge.

Note that in this paper we do not concentrate on the types of features and their "journey" in the development process, as this was already well described in Heikkilä et al. (2013). Moreover, the challenges of transformation and solutions to these challenges concerning fine-tuning of agile shown from the employees perspective are tackled in the aforementioned publication in the qualitative manner (survey based).

4. Goal, questions and metrics for the transformation metrics model

In this section we describe the application of the GQM (Basili et al., 1994) to create our metrics model. We derive the goal and questions from the collaboration with industry partners, progress with literature survey and finally identify metrics for our model.

4.1. Goal

The general goal of transforming and improving development operations was identified in a project-wide survey in 2011. It was further discussed and defined in 2011 within the case company and subsequently iterated to clarify the goal coordinates *issue, object* and *viewpoint* as described by Basili et al. (1994).

The beginning stage of defining the measurement goal consisted of a workshop day with open discussions, held in a world café The World Café format. The GQM *coordinate issues* of the goal, **business value delivery** and **efficiency** were extracted to be of key importance. The workshop participants represented many different roles in the organization, such as: Scrum masters, developers, testers, product owners and line managers. The last coordinate issue, **end-to-end lead time**, was at the time of the workshop counted as part of efficiency, but was later discovered to be more important and subsequently separated. The final GQM goal coordinates are presented below:

Purpose: Improve
Issue: End-to-end lead time, business value delivery and efficiency
Object: The software development process
Viewpoint: From the whole organizations and customer viewpoint

The iterations and refinement of the goal coordinates were finalized during fall 2011 resulting in the following goal:

Goal: *Improve end-to-end lead time, business value delivery and efficiency for the software development process from the whole organization's and customer's viewpoint.*

When improving the lead time, business value delivery and efficiency, there is a risk that the quality suffers. In order to ensure that this has not been the case, we also added quality as a characteristic to measure.

4.2. Questions

Based on the goal and aforementioned coordinate issues, four questions were proposed initially: (1) *Are we more responsive in the new way of working?*, (2): *Do we have better throughput in the new way of working?*, (3) *Do we have a better workflow distribution in the new way of working?* and (4) *Do we have better product quality in the new way of working?*. On a later stage of the study, the wording of the questions was refined in order to remove the positive bias. The final questions are as follows:

Q1: *How did the responsiveness change?*
Q2: *How did the throughput change?*
Q3: *How did the workflow distribution change?*
Q4: *How did the product quality change?*

4.3. Literature survey

A literature survey was performed to map the current status of Software Process Improvement (SPI) evaluation and agile metrics. The literature research was performed by the researchers in October 2011 as an on-line search in the following collections: SpringerLink Springer, IEEE Xplore IEEE, ACM Digital Library and Agile Journal Agile Journal.

The search strings used were: Lead-Time AND Lean, Lead Time AND Lean, Business Value AND Lean, Business-Value AND Lean, Metric AND Lean, Metrics AND Lean, Metrics AND Agile, Metric AND Agile. From roughly 70 articles displayed in the search, 19 articles were chosen as relevant for our metrics study. Furthermore, interesting references from these publications were further examined resulting in a big matrix of publications juxtaposed with their goals and metrics.

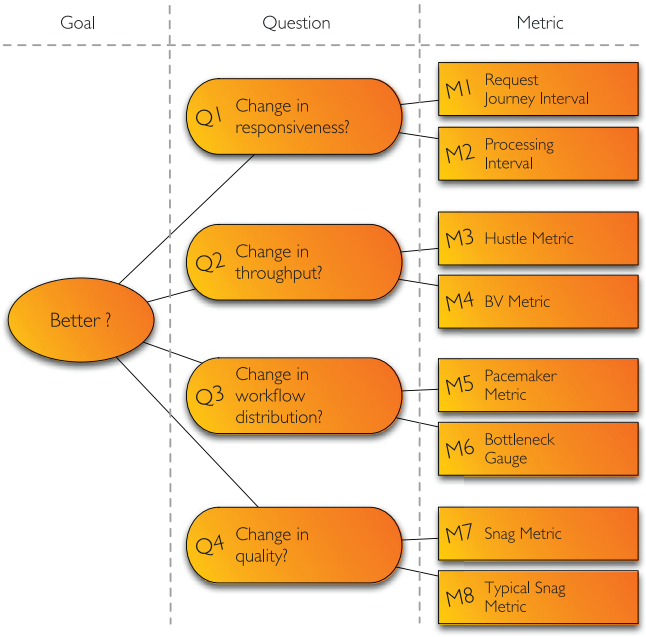
In addition to the concrete metrics discovered in the literature survey, many best practices for agile measurement were also identified in literature (Hartmann and Dymond, 2006; Dubinsky et al., 2005; Hazzan and Dubinsky, 2008). Finally, the candidate metrics were selected for the metrics model.

4.4. Metrics

The final step in the GQM modeling was to determine the metrics to be used. According to Basili et al. (1994) one important factor in choosing metrics is to maximize the use of existing data

sources. The use of existing data sources was particularly important in the research setting since the researchers worked separate from the measured organization and the transformation was already taking place. Subsequent workshops with industry representatives strengthened the potential use of some metrics, whereas other metrics proved difficult to be collected from, e.g., old projects and had to be abandoned.

The metrics model can be seen as a combined performance and quality model, since the attributes defined in the metrics model are implicitly related to performance but also quality attributes (ISO/IEC, 2010). The final metrics model is presented in the following figure, whereas each metric is described in detail in Section 5.



5. Description of metrics

We concentrate on software process quality metrics and measurements which are relevant for both agile and plan driven settings. In the remainder of this section we describe in detail, and provide the reasoning behind, the choice of software metrics and measurements for our model. For each metric we follow a descriptive structure which is a merge of the framework for evaluating metrics presented by Kaner and Bond (2004) with property-based software engineering measurement given by Briand et al. (1996). As a result of this, we provide a detailed and concise description of the purpose, scope and attributes of measurements, as well as the variability and metric formula—all included in this Section, whereas measurement scales, possible relations between metrics and attributes, mathematical properties and dimensional analysis is given in the Appendix. The discussion regarding side effects of using measurement instruments, measurement errors and validity concerns is presented separately in Section 7.

5.1. Q 1. How did the responsiveness change?

With this question we want to explore if there has been an observable change in response time. This question refers to differences in end-to-end lead times in the before and after transformation periods.

Responsiveness is often considered as a key issue in software development. Reinertsen (2009) claims that in domains where response time is vital, this is the only metric that should be utilized for improving service. It is especially important when it comes to supporting operations where solving bugs and problems quickly is of high value to stakeholders. Petersen discusses numerous disadvantages with long lead-times, further motivating measurement and improvement of lead-times (Petersen, 2010).

Likewise, during development of new features, fast lead-time is significant for many competitive benefits, e.g. fast feedback loops and reducing the risk of requirements becoming outdated (waste) (Cockburn, 2006).

Metric 1. Request Journey Interval (Customer Service Request (CSR) turnaround time)

Purpose The metric measures the turnaround time for customer service requests. Customer service requests include the needs issued by the customers for new (or extension of) features, functionality, services (e.g. support), as well as fixing some issues (e.g. bug reports). The metric is calculated as a time period, from a timestamp when the request first comes to the development organization to a timestamp when the request is resolved. These values/timestamps can be collected from the entire duration of the project or from a limited period of time that is of interest. The metric facilitates private self-assessment and evaluation of the organizational process improvement. It brings a higher level viewpoint on the status of customer requests, which serves as informational measure on how well the organization deals with incoming CSRs.

Scope The CSR turnaround time can be measured across organization and/or within a team of an organization for a period of time defined by the organization. It can also be limited by the severity (or priority) and the type of the CSRs.

Attribute Time period.

Variability (attribute) The duration may vary, depending e.g. from the priority, complexity of the task or the organizational capacity to solve the CSR.

Measuring instrument $CSR_{Period} = CSR_{Solv} - CSR_{Cre}$, where CSR_{Solv} and CSR_{Cre} are timestamps for CSR Solved and CSR Created respectively. The measurement is timed (to obtain the measures CSR_{Solv} and CSR_{Cre}) and counted. For the sake of sensible discussion about this metric, we assume that the CSR_{Solv} is later in time than CSR_{Cre}, which means that $CSR_{Solv} > CSR_{Cre}$, where ">" means "older than". Furthermore, we define CSR_{Cre} and CSR_{Solv} as having values that are positive or equal to zero.

Metric 2. Processing Interval: Lead-time per feature (end-to-end)

Purpose The metric measures turnaround time for features selected for development. It is calculated from a timestamp when the feature is accepted for implementation (T_{Impl}) and timestamp when the feature is ready to be shipped (T_{Ship}). Quick turnaround time is essential for competitive advantages as noted by Petersen (2010). The metric aids self-assessment and evaluation of the organizational process improvement. It provides an explicit overview on how long it takes for a feature to be implemented, which gives an informational measure on how well the organization processes the feature (end-to-end). It also endorses metric 4—BV Metric, since shorter turnaround time supports the concept of more frequent releases.

Scope The lead time per feature can be calculated on the organizational and/or a development team level for a given period of time. It can also give an idea of the complexity of features and effectiveness of units working on the features.

Attribute Time period.

Variability (attribute) The length of the time periods may be different, due to differences in the intricacy and priority of the task or the organizational capability to develop a feature.

Measuring instrument LdTime = $T_{Ship} - T_{Impl}$, where T_{Ship} and T_{Impl} are timestamps denoting feature shipping ready date and feature accepted date respectively. The measure is timed (to obtain the measures T_{Ship} and T_{Impl}) and counted. In order for the discussion on metric is reasonable, we assume that the T_{Ship} is later in time than the T_{Impl}, which means that $T_{Ship} > T_{Impl}$.

5.2. Q 2. How did the throughput change?

While the first question concerned the issue of timeliness, this one aims to explore whether the total amount of value delivered changed in the new way of working during similar time periods and projects. The benefits of increased throughput have been discussed widely, including Anderson (2004) and Petersen and Wohlin (2011), therefore investigating this characteristic in the scope of evaluating a transformation is of interest.

Metric 3. Hustle Metric: Functionality/Money spent

Purpose This metric measures how much functionality (also denoted as product size Hazzan and Dubinsky (2008)) can be delivered with respect to a certain work effort. The proposed metric could be computed as the ratio of test points, as described by Dubinsky et al. in Dubinsky et al. (2005); Hazzan and Dubinsky (2008), function points or use cases divided by total money measured in monetary value, or time spent on development measured in person hours. In this work, due to the fact that function point data was not possible to collect retrospectively, we use number of features per money spent. Functionality is defined as single or grouped features, which are possible to sell to a customer. This metric (similarly to Metric 1, Request Journey Interval) also supports BV metric (Metric 4), since more functionality can be split into more frequent releases. The metric supports internal and external assessment and evaluation of the organizational process improvement. It gives a perspective on the cost or work effort needed to implement and deliver certain functionality.

Scope The "amount of functionality" using money spent or specific work effort can be measured across organization and/or within a team in the organization. It can also provide a viewpoint on how effective the work is, which can be used for assessment of organizational improvement with respect to the organizational performance.

Attribute Throughput.

Variability (attribute) The number of the features per money spent may vary, depending e.g. from the complexity, "amount of functionality" to be implemented, or the organizational competence and resources to implement this functionality.

Measuring instrument $\frac{Features}{Money\ spent}$, where features and money spent are natural numbers, both greater than zero.

Metric 4. BV Metric: Nr. of releases/Time period

Purpose Business value is measured as more frequent major releases (Cohn, 2009; Hartmann and Dymond, 2006) in relation to the time period we are interested in (e.g. number of months). The metric aids assessment of the value that the organization generates when developing a set of features or components. The value should be understood not only in a monetary sense, but also in perspective of satisfying customer needs etc. It can be used for organizational process improvement by comparing several samples taken at different points of time. More frequent releases signify keeping up the pace with the needs of users and staying competitive on the market. Moreover, the more frequent releases (which mean rolling out high-value software more rapidly), the quicker value is realized and the risk is reduced (Hartmann and Dymond, 2006). Additionally this metric gives an informational measure on how well the organization processes their tasks, which can act as input for their continuous improvement process.

Scope The business value can be measured across organization and/or within a team of that organization. It can provide a viewpoint on how smooth and well organized the work is, which indirectly transfers to creating business value.

Attribute Throughput.

Variability (attribute) The business value in this case is defined as a relation between major releases and time required for the releases. The variability depends to a large degree from human factor (capability and effectiveness of a team) and the amount of and complexity of the features included in the release, as well as market needs and feasibility of features that can be "packed" in one release in order to create value.

Measuring instrument BV = Number of major releases during a fixed time period, where number of major releases and time period is a natural number. BV equals zero when there has been no releases done in the given time period. For the simplification purposes, we assume that if there were no major releases and there is some development time involved, the value of BV is still equal to zero (although according to the common viewpoint, the profitability would be negative—we have to invest in developers, work environment, etc. and have no chance of getting a return on this investment).

5.3. Q 3. How did the workflow distribution change?

By answering the third question concerning workflow distribution, we want to characterize the way of working, with respect to its iterations. Having frequent iterations enables proactive way of working, meaning discovering the development issues and timely reacting upon them. This is also one of the goals for an agile and lean transformation. Measuring the workflow aids the organization to determine that there has been actual change in the way of working.

Metric 5. Pacemaker Metric: Commit pulse

Purpose Commit pulse measures how continuous integration is done during development by counting the frequency of number of days between commits. The key idea is to keep the frequency and number of days between commits as low as possible, so that the integration is as continuous as possible. The metric is adapted from Dubinsky et al. (2005); Hazzan and Dubinsky (2008), where it is defined within sprints and by counting the number of check-ins per day. In order to for our metric to work for both plan driven and agile settings, we cannot use the term of sprint in our definition. Hence, in our case we scrutinize the data with respect to a larger time-frame, concentrating on visualizing the steadiness of development.

The check-in data can be visualized in a diagram with days between the commits on the x-axis and the frequency of their occurrence on the y-axis (for a defined time period). Optimally, we would have a left hand side shifted bar chart implying few days between commits. The unwanted situation is when the number of days between commits are shifted to the right, which means long periods with no integration.

The metric serves for self-assessment and self-organization purpose. It describes how systematically the workload is distributed throughout the development. It is an informational

measure of how to manage continuous and regular development, in order to avoid integration related issues and risks. Moreover, it reflects the stepwise and steady introduction of changes, which reduces the complexity of integration and decreases the pressure related to issues of meeting the deadline (which can happen with big bang development).

Scope The commit pulse can be measured for a particular time period on the team-level or unit(organization)-level.

Attribute Regularity.

Variability (attribute) The number of days between commits may differ, depending e.g. from the difficulty of the task and the distribution of work. The computation of the metric also differs when e.g. weekends and holidays are not excluded.

Measuring instrument Number of days between commits. The measure is timed and counted.

Metric 6. Bottleneck Gauge: Flow Metric

Purpose The metric measures the difference between the timestamps of each of the handover for features or service requests selected for development. This may be counted either phase-wise (requirements, specification, implementation, testing, deployment) or feature-decision wise (rough idea for the feature, feature concept study, feature implementation, marketing of the feature, including the feature in next release). Having a short handover time is essential for competitive advantages (as noted by Petersen and Wohlin (2011) for turnaround time, which is similar to a concept of handovers). Moreover, it supports responsiveness, which connects this metric back to the first questions. Having a continuous and smooth flow without bottlenecks allows the development organization to quicker respond to customer requests. The metric aids self-assessment and evaluation of the organizational process improvement. It brings an explicit overview on how long it takes for the feature to be implemented, which gives an informational measure on how well the organization processes the feature (end-to-end). It also supports Metric 4—Business value, since shorter turnaround time makes more frequent releases easier.

Scope The timestamps between the handovers per feature can be measured across an organization and/or within a team of the organization for a period of time defined by the organization. It can also give an idea of the complexity of features and effectiveness of units working on the features.

Attribute Time period (handover time).

Variability (Attribute) The length of the time periods may vary, with respect to the complexity and priority of the task, as well as the capabilities of the organization to tackle a feature.

Measuring instrument

HandoverTime = TF_i - TF_{i-1}, where TF_i and TF_{i-1} are timestamps denoting certain phase of processed feature and the proceeding phase of processed feature, respectively and i is a natural number and $i \geq 1$. The measure is timed (to obtain the measures TF_i and TF_{i-1}) and counted. In order for the discussion on metric to be reasonable, we assume that the TF_i is later in time than the TF_{i-1}, which means that $TF_i > TF_{i-1}$.

The metric can be represented by flow diagrams.

5.4. Q 4. How did the quality change?

The three previous questions concern the changes in the development process, whereas this question takes into consideration the quality aspect of the product developed. While we are interested in the transformation and related process differences, the product quality still remains a priority, both for the organization and the customers.

Metric 7. Snag Metric: Number of External Trouble Reports (TR)

Purpose External trouble reports are defect reports submitted from external users. This comparative metric measures the total number of external trouble reports during a certain time period in a release of software in the old way of working compared to total number of external trouble reports from a similar project and similar time period in the new way of working. The metric gives a perspective on the quality of the delivered features or components by counting the number of trouble reports submitted by external users. It additionally gives a before and after view on how many trouble reports were denoted during specific time intervals in development, which can give evidence on the improvement or deterioration of the development process in an organization. It aids the evaluation of a development process and can be used as an indicator for answering the question "are we going in the right direction?" with our organizational performance.

Scope The number of external trouble reports can be measured across organization and/or within a team of that organization for a defined period of time.

Attribute Amount (a number of).

Variability (attribute) The number of External Trouble Reports within periods and in comparison of the old and new way of working may vary, depending e.g. from the quality of the released feature or component and the length of the chosen period. The severity and type of trouble report is not considered in this metric.

Measuring instrument Number of external TR's originating from a certain release is time constrained. The measurement is straightforward and can be directly obtained from objective data sources (e.g. logs, databases, etc.).

Metric 8. Typical Snag Metric: Average Number of Days Open External Trouble Reports

Purpose The metric measures the average number of days, when external trouble reports have had the unsolved status, i.e. from creation of trouble report until it being solved in a given period. It indicates the improvement or deterioration of performance regarding fixes and answers to the received trouble reports. Moreover, it shows whether the organization has actually improved in their way of working, as it brings an explicit overview on how long it takes for the trouble report to be dealt with. It gives an informational measure on how well the organization processes the trouble reports.

This metric is related to responsiveness, but implicitly, it also measures the quality of the product. Thus, in case the trouble reports consistently take longer to solve, then it is likely that the defects found are more complex or that the code base is more difficult to maintain. Furthermore, if trouble reports remain unsolved and there are new ones coming, the overall quality can be considered to have deteriorated. Finally, long response time (number of days open) for external trouble reports may also mean that there are not enough resources to effectively deal with the existing trouble reports.

Scope The number of days to solve the external trouble reports can be measured across organization and/or within a team of that organization for a specific period of time. It can also indicate the complexity of issues reported and effectiveness of units working on the features.

Attribute Time period (days).

Variability (attribute) The duration of the periods may vary, depending e.g. from the complexity and priority of a trouble report, the organizational capacity to fix the reported problem.

Measuring instrument

$$D_{ETR} = \frac{\sum_{i=0}^{n} S_{ETRi} - C_{ETRi}}{n} \quad (1)$$

where C_{ETRi} and S_{ETRi} are days when trouble report i was created and solved, respectively and n is the total number of trouble reports in a period that is investigated.

The measure is timed or dated (to obtain the measures S_{ETR} and C_{ETR}) and counted. In order for the discussion on the metric to be meaningful, we assume that the S_{ETR} is later in time than the C_{ETR}, which means that $S_{ETR} > C_{ETR}$, where ">" denotes the ordering relation "older than".

6. Application of the metrics model

We have applied the proposed metrics model to study the agile transformation at the case organization. In this section, we describe the process of data collection and analyze the gathered data.

6.1. Data collection

The data were collected incrementally through discussions with a number of experts in the company. The collection loop consisted of three main iterative steps: (i) the need for data, (ii) clarifications, and (iii) gaining access to the data. Most of the data were raw datasets (collected automatically), whereas some other data were processed either by tools or the experts themselves. We utilized a third degree data selection strategy (Wohlin et al., 2012), since we used already available and compiled data. We also used archival data, therefore the follow-ups with experts in the organization and filtering the data were necessary.

The priority for our study was to create a setting that would have the highest degree of comparability between two sets of data representing both plan driven as well as agile and lean development. Therefore, we propose a time-wise assessment to investigate the before and after transformation artifacts. This type of investigation is objective and reflects the actual time-line of change, as well as makes automatic data collection easier. Another considered option was a feature-driven investigation, which was thought to be more accurate, but was rejected due to challenges with feasible mapping of the artifacts before and after the transformation (e.g. what features are of comparable size and complexity in the old and new ways of working?). Furthermore, we wanted to prevent the bias that would potentially be introduced when choosing the artifacts for the comparison.

The obtained dataset was gathered from multiple sources, covering the time period from 2006 to 2012. Yet the dataset was not complete for all the aspects under investigation (some files contained more restricted dates). Therefore, after further discussions with experts from the organization, we chose the most representative time intervals that would be suitable for comparison, i.e. 2008–2009 for the old way of working, 2010 for the transformation period and 2011–2012 for the new way of working.

We used the following data sources: logs from customer service requests, database of trouble reports, version control tool reports and additional pre-processed data. The data collection was hardly a straightforward activity, since it required several iterations with the experts from the organization to discuss what data is available and representative for the attribute or metric we want to use. Considering these challenges, the gathering of data for the research became an iterative process (dashed lines), consisting mainly of collection-analysis-clarification loops as illustrated in Fig. 1. Typical topics of discussion in these iterations consisted of clarifying abbreviations and anomalies in the data files. All meetings and discussions were documented and served as valuable notes for the data collection and analysis process. The final step of the data collection process was to review and confirm the usefulness of the existing dataset, as well as appropriately updating it before beginning final analysis (solid line).

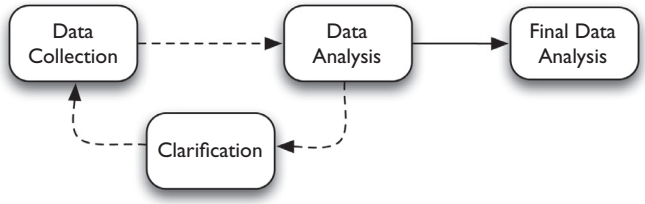


Fig. 1. Visualization of the data collection process.

Most of the data was obtained automatically, by e.g. querying the databases or specifying search criteria in excel sheets. The majority of this data was made available to the researchers in the form of raw data files of considerable size, providing a high level of transparency between the case company and researchers. However, a smaller set of data was pre-processed, e.g. aggregated and normalized with respect to financial entries. The case organization experts graded the data collection effort to be cheap, with most of the effort being spent on e.g. communication and confidentiality considerations. Furthermore, some of the collected and used data was gathered and stored for other purposes than the study; however, it is not available for other parties to be used (non-disclosure agreement).

It should be mentioned that the data availability influenced the choice of metrics and their feasibility (and vice versa) to some degree. This means that other metrics could have been used to better assess certain attributes for agile and lean development, but could not be applied in our case in the plan driven setting (which automatically excluded them from our scope). For instance, the Hustle Metric (metric 3) was initially planned to be measured as number of test points Dubinsky et al. (2005) delivered per time unit, but was altered to number of features per money spent.

6.2. Analysis

The data analysis was done in several iterations, in parallel with data collection as illustrated in Fig. 1. First, the basic and self-explanatory data was collected, studied and the obtained results were presented to the experts. Afterward, more complex, and pre-processed, data was processed and analyzed. Finally, for the results reported here, the complete dataset was analyzed, ran against the metrics we defined, and the results were compared internally at first, then consulted with experts a final time. The data is collected for the period from the year 2008 to the end of year 2012. This large sample size brings additional challenges, like having various tools as data sources, which are further discussed in the validity section.

Regarding the visualizations of the final results on pages 28 - 30, different variations of bar charts were the preferred visual methods to make the comparison between the old way of working (old WoW) and the new way of working (new WoW) as clear as possible despite the data being made anonymous. To aid the comparison, the highest value of each metric corresponds to the numerical value one (1), whereas the other value or values in the same metric are ranging from zero to one (0–1). The scale of y axis is linear. Below we describe our findings with respect to each of the metric from our metric model. Note that Metric 6, Bottleneck Gauge is not depicted here. The reasons are provided in the following section 6.3.

The analysis for Metric 1, Request Journey Interval, was performed directly from a single file of raw data that was made available to the researchers. Dataset reduction was performed for data points that lacked either start or finish time stamp (removing the

incomplete data). The results show roughly a 24% decrease for the customer service request turnaround time going from the old WoW to the new WoW.

Metric 2, Processing Interval, required slightly more effort in the analysis step since the data sources were different due to the transformation. Again, the raw data files were made available to the researchers and through iterations with company experts, the company's development process was discussed and abbreviations were clarified to ensure a meaningful comparison. The data presented consists of actual developed features with representative interrelation on a linear scale. The ordering of the features is randomized. The results show an average decrease of 64% in feature lead-time between the old WoW and the new WoW.

Due to the sensitive nature of the data, a preprocessed data file was supplied to the researchers to compute Metric 3, Hustle Metric. However, the researchers took part in the data collection during a workshop where the correlation between the preprocessed data and the metric was determined. Also in this case, dataset reduction was necessary, this time regarding releases for which the metric $\frac{Functionality}{Money\ spent}$ was impossible to obtain. A scatter plot is used for the data presentation to both show the interrelation between the data points, as well as illustrating the time factor in the metric (showing trends). The average number of features per money spent is 483% higher in the new WoW than on the old WoW. We reason that in the old WoW there was a single, big release scheduled per long period of time, where the amount of functionality was fixed, thus making the releases infrequent. Therefore, there is only one data point representing Hustle Metric in the old WoW. Infrequent releases could have been caused not only by having monolithic type of marketing of features or product packaging, but also by e.g. complexity issues due to the interrelations between features to be deployed. Moreover, the fact that development of one feature might had to wait for other feature to be completed could have disrupted the development flow in the organization. In the new WoW the short iterations impacted the development of certain functionality, since the features to be included could be decomposed into smaller ones and the decisions on what should be included in such "package of functionality" could be made proactively. As a consequence, more frequent releases could be made offering a consumable service or product.

The data for Metric 4, BV, was retrieved in the same workshop as for the Hustle Metric, consisting of a single data file with preprocessed data. The analysis of the data was straightforward with no dataset reduction needed. The results show an improvement of 400% the number of releases during a time period when comparing the old WoW to the new WoW. The reasoning behind this result is the same as for Metric 3, Hustle Metric.

Metric 5, Pacemaker Metric, was analyzed from a single file of raw data extracted from a version control system that was made available to the researchers. We kept the granularity of obtained data on the level of days (as opposed to seconds between commits) in order for the data from the old WoW to be comparable with the ones from the new WoW. In the analysis step, dataset reduction was applied, so that the weekends are not included in the computations. The data is visualized with a bar chart, depicting the frequency of number of days between commits, with separate series for the old WoW and the new WoW. The results show that the maximum days between commits decreased from 12 down to 4. For the remaining days (1–4 days between commits), the occurrences decreased on average by 38%.

The analysis for Snag Metric (Metric 7) was based on a single data file of raw data, which was provided by the case organization. Dataset reduction was performed to conform with releases pertaining to the old and new WoW, respectively. The results show an 188% increase in Trouble Reports. As this metric showed a degradation in performance, the decision was taken to also add data from the transformation period to evaluate the trend for s. The metric can be seen as indicator of product quality for the changing development setting, since it focuses on defect related measurements (in contrast to metrics 1-6, which measure delivery in terms of throughput, responsiveness, etc.). However, Snag Metric does not give self-explanatory results. One could argue that having plan driven development means producing higher quality products within longer time, whereas agile development concentrates on producing cheaper and faster, but at the same time lower quality products. When analyzing the increase of defects one needs to consider that having shorter iterations implies not only delivering more code, but also inevitably increasing the number of possible defects. However, at the same time it increases the likelihood of timely detecting these defects and handling them accordingly.

Metric 8, Typical Snag Metric, was analyzed based on the same data file as Snag Metric with the same dataset reduction. The results for the average time that external Trouble Reports remain open show a decrease of 31%. This metric is tightly connected to Snag Metric and gives context to the interpretation of the results provided above. It actually shows that the detected defects are being handled in a more timely manner, thus contributing to the quality of the product.

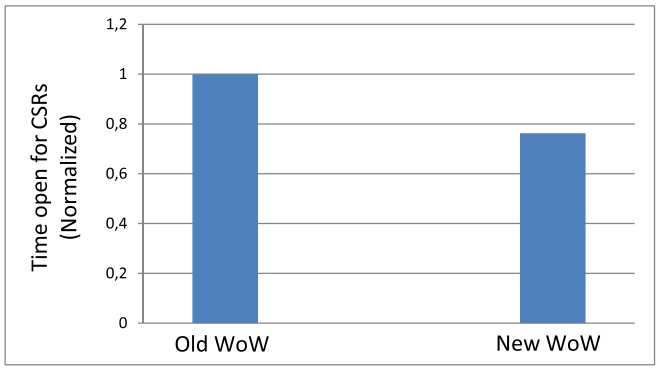
One should keep in mind that the results of application of our metrics are presented in a normalized way, so that the confidentiality of data is preserved. The application of our metrics model to a dataset from a large-scale organization serves as a real-life example to illustrate the usability of our approach; however, it should not be considered as our main contribution.

6.3. Discussion on the collected measurements

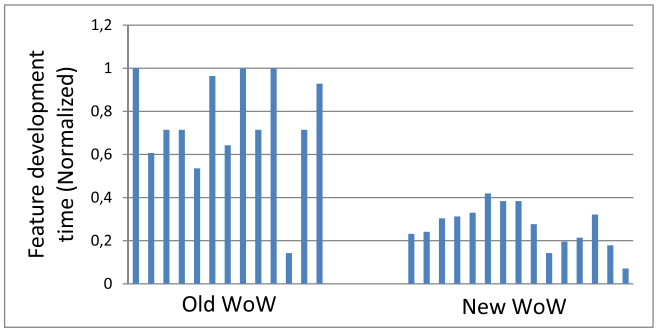
Our results show that the most significant improvement occurred in the change in throughput (Q2), both considering developed and deployed functionality per money spent (M3) and the number of releases in a certain time period (M4). The improvements were 483% and 400% respectively. The second largest improvement was observed in faster responsiveness (Q1), both to process a customer service request (M1) with a 24% decrease and to develop a feature (M2) showing a 64% decrease in time. Moreover, the time between commits has shortened 38% (M5), which signifies that the working code is sent to the repository more often, which by itself smoothens and speeds up the development. Furthermore, the quality aspect of development in the organization improved (Q4), as the time necessary to answer and tackle reported problems decreased with 31% (M8).

Interestingly enough, number of external trouble reports increased with 188% (M7). When looking at the data from the transformation period, a decreasing trend can however be identified. A reason for this change in trouble reports may be due to more functionality being implemented (M4) and delivered in shorter time (M2). As a consequence, there are more potential customers who provide more feedback and can send a trouble report with same problem multiple times, causing a sudden rise in the number of trouble reports.

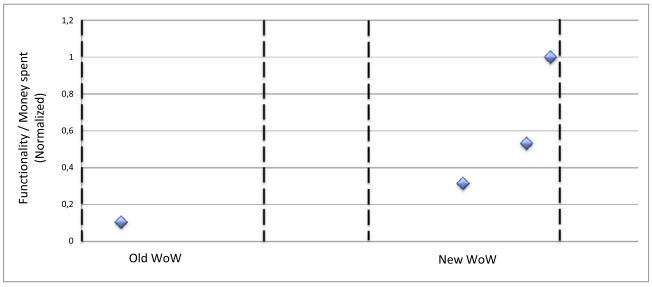
The analysis of the old WoW showed that the organization was tackling tasks very well and smoothly, without "empty" waiting times, when the task needed to be finished and handled from the earlier phase. The experts from case company explained that the development was not on hold in this scenario; rather, the developers were switching between tasks. Thus the changes in the development process in new WoW caused the handover time between the decision of taking a feature into the analysis for the development, acceptance/rejection for development and placing it prioritized to product the backlog, and finally implementation and marketing of a feature, was shortening proportionally, also along the time agile/lean was adapted.



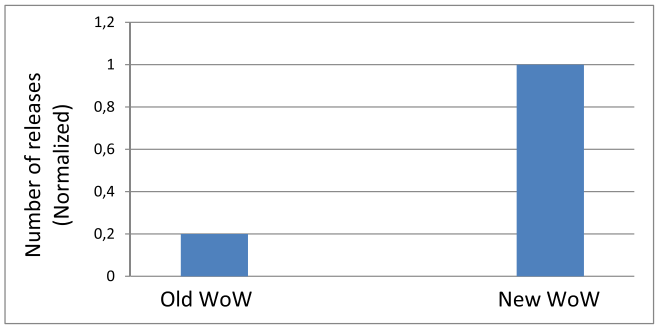
(a) Metric 1 - Request Journey Interval



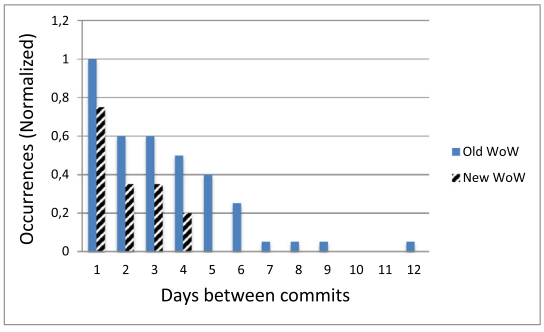
(b) Metric 2 - Processing Interval



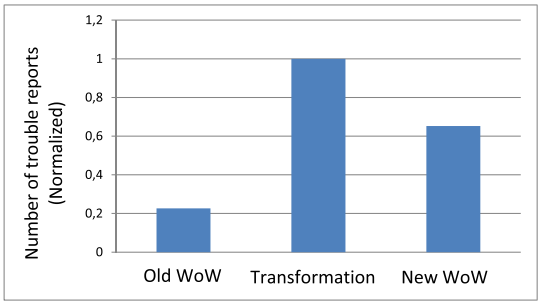
(c) Metric 3 - Hustle Metric



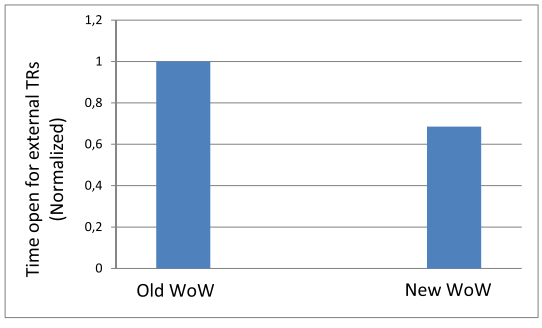
(d) Metric 4 - BV Metric



(e) Metric 5 - Pacemaker Metric



(f) Metric 7 - Snag Metric



(g) Metric 8 - Typical Snag Metric

Only one of our metrics (M6 Bottleneck Gauge) occurred to not be feasible to be applied both in the plan-driven and agile and lean setting in a straightforward manner. This was caused by the dataset that we were able to obtain from the organization. According to our assumption that the metrics should be straightforward to apply and the data should be easy obtainable, we were not able to have a direct comparison of the old and new WoW for the handover time. The data reported in the plan-driven development differed significantly from those in the agile and lean setting. The transformations on the data would consume significant amount of effort, therefore, this practice would be against the practicality and applicability of metrics. Based on our experience, Metric 6 could be more applicable in a transformation made in smaller steps that are retaining more of the old way of working.

As can be seen in Table 1 on page 31, the results are to a high degree noticeable, with two metrics showing 400% and above improvement. The results were discussed with different members of the case company and, whereas the actual numbers were new, they corresponded to the experienced improvement in the company.

Naturally, not all observable changes that were indicated by measurements can be explained solely by the transformation as the case company operates in a rapidly changing market. However we tried to limit the factors impacting the measurements. We purposefully narrowed down the scope to measuring the software development organization, in contrast to the whole company. By that we concentrated on the turnaround times and trouble reports, and thus minimized the effects of external factors such as revenue or user experience.

7. Validation of metrics

In Section 3.1 we listed five criteria used for selecting the metrics of the proposed model. In this section we explain why we consider these criteria to be fulfilled by the proposed metrics. We also consider the validity of the metrics model according to the criteria given by Meneely et al. (2013). Finally, we study the validity of the application of the metrics model in the case organization and point out the limitations of our study.

7.1. Validation of the model with respect to own criteria

7.1.1. C1 and C3: plan driven and agile projects, independent of scope, size and complexity

With the exception of the workflow distribution metrics (Metrics 5 and 6—Pacemaker Metric and Bottleneck Gauge), all the selected metrics measure the development effort from an external point of view. The time and effort for delivering service requests, features, functionality, business value, and trouble reports are measured looking at the point when they enter and exit the development organization. As such they ignore the internal process used to produce the result, and thus render the metrics independent of the internal process model used, i.e. plan driven or agile (C1). As the metrics do not consider the internal workings of the project measured, it is also agnostic to scope, size and complexity (C4).

In contrast, the workflow distribution metrics illustrate the difference between the two process models, in order to verify that a change in way of working has indeed taken place. Metric 5 (Pacemaker Metric) is trivial to collect in any organization using a version control tool for the produced source code, regardless of the process model used. Metric 6 (Bottleneck Gauge) requires that the organization uses some form of time tracking tool for the development activities, and has continued to do so after the transformation. This may not always be the case.

7.1.2. C2: support for agile values

As noted by Hartmann and Dymond, the inappropriate use of metrics can threaten an emerging agile and lean culture Hartmann and Dymond (2006). For this reason, we were careful to select metrics that support agile values (C2). The core values and principles described by the agile manifesto Agile Alliance are centered around responsiveness, early delivery of working software, cooperation and communication, technical excellence, simplicity, self-organization and human interaction. Throughout the selection process, these values have served as a guide.

7.1.3. C4: objective

All the data collected is quantitative and, with the exception of Metric 6 (Bottleneck Gauge), requires no interpretation of experts. The entry and exit dates for implemented items as well as the number of items present should all be objective information present in the documentation of the projects measured. The Flow Metric is a slightly more complex metric that requires plotting and analysis of the plotted curve. This does not compromise the objectivity of the metric, but makes the implications of it slightly more cumbersome to analyze.

7.2. Validity according to Meneely et al. criteria

We studied and followed the validity scheme described by Meneely et al. (2013), where the authors listed 47 validity criteria and thoroughly described their semantics. In our work a subset of 20 criteria (denoted in *italics*) was identified to be applicable: *applicability, mathematical properties, scale validity, unit validity, monotonicity, clear definition, internal validity, internal consistency, transferability, repeatability, accuracy, reliability, non-exploitability, construct validity, relevancy, usability, external validity, economic productivity, constructive, actionable*. The aforementioned criteria give a perspective on the purpose of our metrics and some of their characteristics on a more general level.

7.3. Validity of the application of the metric model in the case organization

Empirical investigations entail a degree of uncertainty regarding the validity of obtained results. Therefore, discussing the validity threats is not only helping to better understand (and possibly replicate) the presented work, but also gives the reader a sense of trust in the presented results. In this section we analyze the validity of the application of the proposed metrics in an actual organization. Note that the large scale example serves as an illustration for our work on metrics for agile transformation. We follow the validity discussion scheme proposed for software engineering domain by Wohlin et al. (2012).

Internal validity In our work we wanted to ensure that the agile and lean transformation caused the observed changes (the effects). Here, we discuss that the changes were not resulting from factors of which we had not planned or measured.

While the selection of subjects had no impact in our investigation (motivation and suitability of persons involved was appropriate), the non-human related factors were affecting the complexity of the study, e.g. impacting the data collection process. The instrumentation, although bringing objectivity, triangulation and automation to our work, entailed some difficulties due to having multiple tools used by various experts providing different format of data. Thus, we needed to make sure that we obtain comparable datasets, which entailed iterating on feasibility of datasets by dataset reduction and discussions with experts from case company.

External validity The main concern in the discussion about external validity is the issue of conducting the investigation in a single organization. However, the context of our work, as well as metrics themselves were carefully described (Sections 3.2 & 5 respectively), which significantly reduced that risk.

The methodology chosen (GQM) produces metrics models that are specific for a given context and therefore make it difficult for us to generalize the results. The presented outcomes can be to some degree generalizable in contexts similar to ours, i.e. large-scale and customer driven organizations, which are either planning, executing or are after the agile and lean transformation (or transformation to an iterative and incremental development processes). We kept the environment as realistic as possible, therefore the application of our metrics model to other industrial practices has potential. The findings are envisaged to provide valuable insight to practitioners and researchers, who are interested in applying the metrics model themselves.

Table 1
Summary of the results obtained from the collected data sets.

Metric	Indicator of success	Result	Interpretation
M1 Request Journey Interval	Decreased time interval	−24%	Improvement
M2 Processing Interval	Decreased time interval	−64%	Improvement
M3 Hustle Metric	Increased functionality per money spent	+483%	Improvement
M4 BV Metric	Increased number of releases / time period	+400%	Improvement
M5 Pacemaker Metric	Decreased interval between commits	−38%	Improvement
M6 Bottleneck Gauge	Decreased HandoverTime	N/A	N/A
M7 Snag Metric	Decreased amount of TRs	+188%	Degradation
M8 Typical Snag Metric	Decreased time interval of open TRs	−31%	Improvement

Naturally, the implementation of certain metrics may differ between organizations, meaning that e.g. the business value or functionality metric may be defined in some other way that takes into account the specifics of that organization (C3 from our criteria for Metrics Model). Moreover, the specifics of an organization create possibility of not obtaining feasible data, which in turns means that some metrics cannot be calculated. This is the reason why we decided to have two metrics supporting each question. In case one metric cannot be obtained due to e.g. incomparable data or lack of data, there is another metric to answer a given question and at the same time achieve C3. Also in our work we have an example of such a scenario: Metric 6 - Bottleneck Gauge occurred to be not applicable in our setting due to the incomparable datasets for the old and new WoW.

Construct validity When discussing construct validity we are concerned if the right measures were used for the concept being studied. The thorough definition of metrics, followed by their validation adds value to the overall construct validity of the investigation. Although there is always a danger that the researcher(s) or the organization may impact the outcome of the investigation, in our case it was reduced by having several rounds of reviews of the work-in-progress done by co-authors of this paper and the representatives of the organization. The common understanding of the metrics used was achieved due to multiple discussions. Finally, the organization had influence on establishing the metrics model only on the conceptual level. Due to confidentiality reasons, the data that were made available to the researchers were impacted by the organization (delivering pre-processed data); however, it did not bias the resulting measurements.

Not only were we careful for the environment (people) not to bias our measurement process, but we also saw to that the measurements made using the proposed metrics did not affect the case organization or people involved in the study. We obtained data from certain periods of time and did not observe the organization in real-time (thus we did not interfere with the processes and practices in the case organization). Naturally, the results of analysis were considered as precious feedback for the purpose of self-improvement and thus supporting agile principles.

We investigated the problem of a “change” from many perspectives, i.e. having a number of metrics, thus we avoid the mono-method bias. Moreover, the experiment setting reflects the construct under study—we are exploring a change in its natural environment with the appropriately defined metrics that were previously studied in research and applied in practice. Furthermore, our metrics have been tailored specifically for the organization described in the empirical study with respect to the available data.

Conclusion validity (reliability) When collecting data there often rises a challenge of the data collection, as well as the completeness and quality of the obtained dataset. In our work we tackle a large dataset, which was reduced due to incompleteness of records, e.g. lacking the start or end dates. The lack of a few data points was compensated by results of other metrics (e.g. functionality could be balanced by the BV metric). Furthermore, some of the data originating before the explicitly defined investigation periods were "cut-off", thus it is not possible to state what impact this data could have had on the outcome. However, we discussed this issue with the experts from the organization and as a result the most feasible periods of time were chosen.

In our work we are not focusing on statistical significance. We are more concerned with the careful data collection and definition of metrics, in order to obtain a meaningful measurements in our study and provide feedback to the case company. We propose metrics model that is generic, but at the same time feasible to be applied to the case organization (the model can be tailored to the specifics of an organization). Furthermore, the data collection and metrics computation is done automatically, thus limiting the human factor in the measurement process. We are not claiming that none of the metrics can be manipulated, but when using automation and having precisely defined metrics this risk is very much reduced. Finally, by neutrally formulating our main research question and the supporting questions, we are not searching for a desired outcome, e.g. improvement. Rather, we are interested in the change itself, regardless if it is for better or worse.

8. Conclusions

Measurements of agile development process and especially the organizational changes leading to establishing this process in large-scale organizations have been neglected in research (Dybåand Dingsøyr, 2008). In this article, we present a metrics model to quantitatively compare a software development organization before and after an agile and lean transformation. Our goal with this work was to provide a quantitative approach as an alternative to qualitative studies such as interviews and surveys.

The metrics were elicited with the use of the *GQM* approach (Basili et al., 1994), which consisted of four questions. For each of these questions there were two metrics established according to three key factors: i) The metrics were to answer the research question about the changes resulting from the transformation from the traditional way of working (plan driven) to agile and lean. ii) They were meant to be comparable, i.e. used, accepted and computable in both the old and new way of working. iii) Finally, the data for the computations were supposed to be available and valid. Depending on the availability of data and rigor of the investigation, it might suffice that only one out of these two metrics is computable in order to be able to answer a given question. The characteristics of interest are reflected in the four questions presented earlier, i.e.: responsiveness, throughput, workflow distribution and quality.

In perspective of an agile and lean transformation, the general purpose of metrics is to use them as techniques to analyze organizational change, with respect to the questions listed in Section 5. Besides being used for comparison purposes, metrics from our measurement model serve to demonstrate the current state of development, encourage self-assessment and facilitate continuous improvement. Moreover, they provide transparency to project status evaluation, as well as act informatively with respect to e.g. (potential) customers whenever necessary.

Metrics are expected to support agile principles by providing tangible evidence to aid collaboration and self-organization in agile teams, rather than serve as control mechanisms (Hartmann and Dymond, 2006). The proposed metrics are not designed to be used for inspecting and comparing the performance of different teams. On the contrary, the teams should be able to react quickly to the development needs themselves.

8.1. Implications of the study

Organizations interested in gaining a deeper understanding of the software development processes dynamics, in particular the observable characteristics between different development approaches, are encouraged to apply the proposed metrics model in their context. Moreover, our metrics model supports the identification of improvement areas in the development process. The presented metrics are thought to be generic in the sense that they are applicable to different settings (e.g. (i) plan driven development, (ii) agile and lean), as well as feasible for the assessments of various types of granularities, i.e. assessing single release, sprint, feature, or collect measurements within certain period of time. Moreover, they can provide a high-level viewpoint on the organization, without specifying any time and resource constraints. In this case they can be treated as indicators for process measurements.

When applying our metrics model to the case company, we showed that the metrics are sensitive to the organizational changes. We also quantitatively confirmed that agile transformation is a journey, which needs continuous monitoring, in order to provide a status check and indicate possibilities of improvement. We also demonstrated that the change of the development process should entail special attention for quality aspects of development (see Metrics 7 and 8, Snag Metric and Typical Snag Metric). Moreover, our metrics model clearly showed that providing a single quality metric may obscure the overall picture given by measurements. The (seeming) deterioration of quality in terms of increase in Trouble Reports (see Metric 7, Snag Metric) alone was compensated by the decrease in time for the defect to remain unfixed (see Metric 8, Typical Snag Metric). Thus it is advised that the measurements of Metric 7 should be analyzed together with the ones for Metric 8.

Monitoring of responsiveness and throughput, characteristics that are the cornerstones of agile approaches, is well supported by our metric model. We emphasize that checking performance by applying our metrics model should be done without the judgmental flavor, i.e. the measurements should serve as information radiators, rather than be used as a “plunger of blame”. Moreover, they are to be utilized for promoting the responsibility among team members and help to timely identify bottlenecks.

It is quite likely that some of the metrics proposed in the metrics model will require fine-tuning to the specifics of the application setting. For instance, the Business Value metric (Metric 4) can be defined in some other terms for various organizations. In this paper we defined it on the basis of the descriptions provided by experts from the case organization as number of major releases for certain time period. However, it can be computed otherwise by e.g. allocating business value points to a minimally marketable feature and then tracking the completion of business value points.

An example of re-working the metrics according to application setting, availability of data and comparability of datasets that is present in our work is Bottleneck Gauge, Flow Metric (Metric 6). When developing this metric, we were inspired by the work of Petersen and Wohlin (2011). For the old WoW we were able to obtain data for metric 6 in accordance to the original description of the flow metric (Petersen and Wohlin, 2011). However, we could not obtain corresponding data for the new WoW. Therefore, the current description of flow metric based on HandoverTimes was developed after discussions between the researchers and company representatives. However, the initial analysis of datasets for old and new WoW showed that they were not comparable. Although a method for converting the old WoW data to correspond to the new WoW data format was proposed, it was considered too expensive to perform and would violate the properties of the metrics model for the measurements to be practical and easy to obtain. Ultimately, the metric was not applied in the case organization. Its description is purposefully left and discussed in this paper in order to serve organizations that e.g. carry out a transformation in smaller steps.

Further implications of our work will be known once we experiment with more case studies, see Section 8.2.

8.2. Future research directions

So far we applied our metrics model to one large-scale organization; however, we plan further experimentation with our theoretical work by utilizing our model in other large-scale organizations, regardless of their domain. We are particularly interested in investigating which metrics need to be fine-tuned, how much effort is involved in adjusting the metrics and if this kind of modifications are at all feasible.

We would like to qualitatively explore which of the proposed metrics are the most beneficial for which groups in the organization, taking into account the task orientation of the teams. Finally, we would like to examine if there emerges a need for additional metrics, which would extend our model.

Acknowledgments

This work was supported by the DIGILE project Cloud Software Finland. The work of first author was partially funded by the Academy of Finland project ADVICeS (no. 266373).

Appendix

In this appendix we list the metrics proposed by our metrics model and describe their formal properties, which were not included in the main publication.

Metric 1. Request Journey Interval (Customer Service Request (CSR) turnaround time)

Natural scale (attribute) Ratio scale (although timestamps are interval scale).

Natural scale (metric) Interval scale, since two elements of the metric are measured on the interval scale. Moreover, it is possible to make measurements from an arbitrary epoch in time (assuming that the data exist). There also exists a ”zero point”, which is arbitrary and to be defined by the organization applying the metric. However, negative values cannot be used, since it would indicate that the definition of the metric is false.

Attribute–metric relationship The relationship between attribute and metric is straightforward and self-explanatory.

Formal properties non-negativity and positivity (self explanatory), zero-element—immediate closing of CSRs without an effort needed, disjoint attribute additivity meaning that if one CSR is solved in certain period of time, second not related CSR is solved in another period of time, one needs to have the sum of periods to solve both CSRs; monotonicity, as the time period always increases or remains constant as the time and measurement process progresses; normalization is possible as the comparisons between measured attributes are meaningful, since they all belong to the same interval.

Dimensional analysis We recommend using days as a unit of measurement, but in case of organizations, where CSRs are solved in significantly shorter time intervals, it might be beneficial to use, e.g. hours.

Metric 2. Processing Interval: Lead-time per feature (end-to-end)

Natural scale (attribute) Ratio scale (however timestamps are interval scale).

Natural scale (metric) Ratio scale, although two elements of the metric (timestamps) are measured on the interval scale. Furthermore, measurements can be obtained from an arbitrary period of time (if there are data points for this period). A specific "zero point" should be defined by an organization. Nevertheless, no negative values can be used, as it would be against the definition of relation between timestamps (elements of metric). Moreover, it is meaningful to say that the implementation of feature A was e.g. twice as long as feature B.

Attribute–metric relationship The relationship between attribute and metric is direct and easy to follow.

Formal properties non-negativity and positivity (straightforward), zero element (feature can be concluded with no effort needed, because e.g. the implementation was trivial or the feature is implemented already as a part of other implementation), disjoint attribute additivity (two disjoint lead times per features is equal to the sum of the lead times); monotonicity (this measure always increases or remains constant as the time and measurement process progresses); normalization (meaningful comparison between lead times per feature—as they all belong to the same interval).

Dimensional analysis Recalculating the metric for different time-granularities (hours, days, etc.) is straightforward and depends on the specificity of the development.

Metric 3. Hustle metric: Functionality/Money spent

Natural scale (attribute) A feature, i.e. a product or a service for which the company can charge the customer—absolute scale (simple count), money spent—ratio scale.

Natural scale (metric) Ratio scale - dictated by the money spent element, since it is represented by a weaker scale than absolute scale (features). There is a "zero point", which is when there are no features or there has been no money involved in implementing the functionality. Negative values cannot be used, since it would indicate that the definition of the components of metric is invalid. Moreover, it is meaningful to say that the implementation of certain functionality (feature A) costs twice as much as some other functionality (feature B), disregarding the fact that the functionality delivered by the features might be totally different and is not comparable.

Attribute–metric relationship The relationship between attributes and metric is valid only if we assume that the measure is equal to zero in case when there are no features or there is no effort required for developing the feature.

Formal properties Non-negativity and positivity (straightforward), zero element (no costs of a development or no features), disjoint attribute additivity (two disjoint features of certain different functionality need as much money to be implemented, as the sum of their functionalities; naturally here we do not consider the design factors and the entailed complexity issues, which might impact the development cost when dealing with larger and more complex features); monotonicity (this measure decreases with respect to the growing amount of money required for the development of a certain functionality); normalization is not meaningful, as the type and complexity of functionality to be implemented and solutions used may differ.

Dimensional analysis In our work we propose a measure of the type: "number of features per money spent". However, we can easily change the "functionality" unit of this measurement by admissible transformation into e.g. features or components. Moreover, money spent can be interpreted as some other resources used, e.g. person hours. Thus dimensional analysis is possible.

Metric 4. BV Metric: Nr. of releases/ime period

Natural scale (attribute) Number of major releases—absolute scale (simple counting), time period—ratio scale.

Natural scale (metric) Ratio scale, although two elements of the metric number of major releases and time period are measured on the absolute and ratio scales, respectively. It is possible to make measurements for a given epoch in time (assuming the presence of data). There is a "zero point", which is denoted by the beginning of the work, where there is time frame yet and there are no releases available. Negative values cannot be used, since the number of releases and work effort have a non-negative value.

Attribute—metric relationship The relationship between attribute and metric needs some additional assumptions for the computations (see the above description). Additionally, it should be noted that the metric is more oriented toward organizational performance and market sustainability, than for instance purely monetary value.

Formal properties Non-negativity and positivity (from the definition), zero element (no releases made), disjoint attribute additivity exist in theory (two disjoint business values are equal to their sum) although is not applicable in reality (scenario when two disjoint releases combined have more value for the user than separately); monotonicity (BV measure decreases with respect to the growing amount of time required for the development of a release); normalization is not meaningful, as the type of features included in the release may differ.

Dimensional analysis This measure is of the type: "releases per time period", where releases should be defined by the organization. The "time unit" of this measurement can be changed by admissible transformation (e.g. between hours, days, months or years, depending on the development characteristics). Thus dimensional analysis is possible.

Metric 5. Pacemaker Metric: Commit pulse

Natural scale (attribute) Absolute scale (simple counting is the only possible measure).

Natural scale (metric) Absolute scale. Decimals are allowed, however negative values cannot be used.

Attribute-metric relationship The relationship between attribute and metric is straightforward, i.e. the data collected for the number of days between commits for a specific time frame will provide us with data points for a certain period. The data set can be plotted with a bar chart of a frequency of number of days between commits.

Formal properties Non-negativity and positivity (straightforward), zero element (no commits done), no disjoint attribute additivity (two disjoint numbers of days between commits is not equal to the sum of the days between commits); normalization (meaningful comparisons between the number of days between commits per time period).

Dimensional analysis Units of measurement are simple counts and are fairly straightforward in this case. We suggest using days between commits as a unit of measurement. If the

commits are more frequent, as desired in agile processes, we suggest changing the granularity from days to hours.

Metric 6. Bottleneck Gauge: Flow Metric

Natural scale (attribute) Ratio scale.

Natural scale (metric) Ratio scale, even though timestamps are on the interval scale. Additionally, assuming that the data exists, measurements can be made from a specific time period. Furthermore, a "zero point" can be identified specifically by the organization. However, negative values are forbidden to be used (indication of a false definition of relation between timestamps). Finally, saying that the implementation of feature A took e.g. twice as much as for feature B is meaningful.

Attribute–metric relationship The relationship between attribute and metric is direct and easy to understand.

Formal properties non-negativity and positivity (straightforward), zero element (no work performed on a feature), disjoint attribute additivity (two disjoint handover times per features is equal to their sum); monotonicity (this measure always increases or remains constant as the time and measurement process progresses); normalization (meaningful comparisons between handover times per feature—as they all belong to the same time-frame).

Dimensional analysis The granularity of the time measurements should vary according to the context (specifics of organization and development). Recalculation of metrics from days to hours is rather basic.

Metric 7. Snag Metric: Number of External Trouble Reports (TR)

Natural scale (attribute) Absolute scale (one measurement mapping—a simple count).

Natural scale (metric) Absolute scale, it is a number of occurrences of the external trouble reports. Can be obtained from a specific time-frame (if there is data). There also exists a "zero point", meaning no external trouble reports. The measurement mapping starts at zero and increases in equal intervals (units). In order for the metric to make sense, negative values cannot be used. Moreover, for the comparison reasons, it is meaningful to relate the number of external trouble reports in old way of working with the new way of working, i.e. it is meaningful to say that the number of external trouble reports in old way of working is bigger (or twice as big) as in the new way of working.

Attribute–metric relationship The relationship between attribute and metric is self explanatory.

Formal properties Non-negativity and positivity (straightforward), zero element (no external trouble reports), disjoint attribute additivity (two disjoint numbers of external reports, where disjoint means that the period from where the measurements are taken does not overlap, is equal to the sum of the numbers of these external reports); monotonicity (increases or remains constant with respect to the progress of time); normalization (meaningful comparisons between external trouble reports—as they all belong to the same interval).

Dimensional analysis Assignment of units of measurement is rather straightforward in this case, it is a simple count. There is no need to change the unit of measurement in this case.

Metric 8. Typical Snag Metric: Average Number of Days Open External Trouble Reports

Natural scale (attribute) Absolute scale (one measurement mapping—a simple count of days).

Natural scale (metric) Ratio scale, although two elements of the metric (dates) are measured on the interval scale for a particular period of time. The organization is the one to identify a "zero point", from which a measurement starts. No negative values can be used. Moreover, it is meaningful to say that solving the external trouble report A took e.g. twice as much or longer than for trouble report B (severity of tackled task is not in the scope of this metric).

Attribute–metric relationship The relationship between attribute and metric is based on the basic mathematical computations (computing average), which are allowed in the ratio measurement scale.

Formal properties non-negativity and positivity (straightforward), zero element (no work performed on the trouble report), disjoint attribute additivity (separately solving two disjoint error reports is equal to the sum of the time it takes to solve them); monotonicity (this measure always increases or remains constant as the time when trouble report is open progresses); normalization (meaningful comparisons between days when reports are having the open status).

Dimensional analysis Computing the metrics for various time-granularities is straightforward.

References

ACM Digital Library. www.dl.acm.org/.

Agile Alliance. Agile Manifesto. www.agilemanifesto.org/.

Agile Journal. www.agilejournal.com/.

Anderson, D.J., 2004. Agile Management for Software Engineering: Applying the Theory of Constraints for Business Results. The Coad Series. Prentice Hall.

Basili, V.R., Caldiera, G., Rombach, H.D., 1994. The goal question metric approach. In: Encyclopedia of Software Engineering. Wiley, pp. 646–661.

Basili, V.R., Rombach, H.D., 1988. The tame project: towards improvement-oriented software environments. IEEE Trans. Softw. Eng. 14 (6), 758–773. doi:10.1109/32.6156.

Boehm, B.W., 1978. Characteristics of Software Quality. TRW Series of Software Technology. North-Holland Pub. Co.

Briand, L.C., Morasca, S., Basili, V.R., 1996. Property-based software engineering measurement. IEEE Trans. Softw. Eng 22 (1), 68–86.

Cloud Software Programme, 2013. Cloud Software Finland.

Cockburn, A., 2006. What engineering has in common with manufacturing and why it matters. J. Defence Softw. Eng 20 (4), 4–7.

Cohn, M., 2009. Succeeding with Agile: Software Development Using Scrum, first Addison-Wesley Professional.

Concas, G., Marchesi, M., Destefanis, G., Tonelli, R., 2012. An empirical study of software metrics for assessing the phases of an agile project. Int. J. Softw. Eng. Knowl. Eng. 22 (04), 525–548. doi:10.1142/S0218194012500131.

Deemer, P., Benefield, G., Larman, C., Vodde, B., 2012. The Scrum Primer—A Lightweight Guide to the Theory and Practice of Scrum. Technical Report.

Destefanis, G., Counsell, S., Concas, G., Roberto, T., 2014. Software metrics in agile software: an empirical study. In: Agile Processes in Software Engineering and Extreme Programming. Springer International Publishing, pp. 157–170. doi:10.1007/978-3-319-06862-6_11.

Dingsøyr, T., Fægri, T., Itkonen, J., 2014. What is large in large-scale? A taxonomy of scale for agile software development. In: Jedlitschka, A., Kuvaja, P., Kuhrmann, M., Männistö, T., Münch, J., Raatikainen, M. (Eds.), Product-Focused Software Process Improvement (PROFES). Springer International Publishing, pp. 273–276. doi:10.1007/978-3-319-13835-0_20.

Dingsøyr, T., Moe, N.B., 2013. Research challenges in large-scale agile software development. SIGSOFT Softw. Eng. Notes 38 (5), 38–39. doi:10.1145/2507288.2507322.

Dromey, R.G., 1995. A model for software product quality. IEEE Trans. Softw. Eng. 21 (2), 146–162. doi:10.1109/32.345830.

Dubinsky, Y., Talby, D., Hazzan, O., Keren, A., 2005. Agile metrics at the Israeli Air Force. In: Proceedings of the Agile Development Conference (ADC'05). IEEE, pp. 195–209.

Dybå, T., Dingsøyr, T., 2008. Empirical studies of agile software development: a systematic review. Inf. Softw. Technol. 50 (9–10), 833–859. http://dx.doi.org/10.1016/j.infsof.2008.01.006.

Ebert, C., Abrahamsson, P., Oza, N., 2012. Lean software development. IEEE Softw. 29 (5), 22–25. doi:10.1109/MS.2012.116.

Gilb, T., 1988. Principles of Software Engineering Management. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Hartmann, D., Dymond, R., 2006. Appropriate agile measurement: using metrics and diagnostics to deliver business value. In: AGILE 2006 Conference (Agile'06). IEEE Computer Society, pp. 126–134.

Hazzan, O., Dubinsky, Y., 2008. Agile software engineering. Undergraduate Topics in Computer Science. Springer, Berlin.

Heikkilä, V., Paasivaara, M., Lassenius, C., Engblom, C., 2013. Continuous release planning in a large-scale scrum development organization at Ericsson. In: Proceedings of the 2013 International Conference, XP2013 on Agile Processes in Software Engineering and Extreme Programming. Springer-Verlag, Berlin, Heidelberg, pp. 195–209.

IEEE. IEEE Xplore Digital Library. www.ieeexplore.ieee.org/.

ISO, 2000. ISO 9004:2000, Quality Management Systems—Guidelines for Performance Improvements. Technical Report. International Organisation for Standardization.

ISO, 2001. ISO 9126-1:2001, Software Engineering—Product quality, Part 1: Quality Model. Technical Report. International Organisation for Standardization.

ISO/IEC, 2010. ISO/IEC 25010—Systems and software Engineering—Systems and Software Quality Requirements and Evaluation (SQuaRE)—System and Software Quality Models. Technical Report. ISO/IEC.

Kaner, C., Bond, W.P., 2004. Software engineering metrics: what do they measure and how do we know? In: In Proceedings of 10th International Metrics Symposium (METRICS 2004). IEEE CS Press, pp. 1–12.

Kitchenham, B.A., 1996. Software Metrics: Measurement for Software Process Improvement. Blackwell Publishers, Inc., Cambridge, MA, USA.

Korhonen, K., 2013. Evaluating the impact of an agile transformation: a longitudinal case study in a distributed context. Softw. Qual. J. 21 (4), 599–624. doi:10.1007/s11219-012-9189-4.

Korhonen, K., 2014. Migrating defect management from waterfall to agile software development in a large-scale multi-site organization: a case study. In: Agile Processes in Software Engineering and Extreme Programming (XP2014). Springer, Berlin, Heidelberg, pp. 73–82. doi:10.1007/978-3-642-01853-4_10.

Leffingwell, D., 2007. Scaling Software Agility: Best Practices for Large Enterprises. The Agile Software Development Series. Addison-Wesley Professional.

Li, J., Moe, N.B., Dybå, T., 2010. Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '10). New York, New York, USA.

McCall, J.A., Richards, P.K., F., W.G., 1977. Factors in Software Quality. Concept and Definitions of Software Quality.. Technical Report.

Meneely, A., Smith, B., Williams, L., 2013. Validating software metrics: a spectrum of philosophies. ACM Trans. Softw. Eng. Methodol. 21 (4), 24:1–24:28. doi:10.1145/2377656.2377661.

Mikkonen, K., Seikola, M., Jouppila, M., Christian, E., 2012. How We Learn to Stop Worrying and Live with the Uncertainties? https://www.cloudsoftwareprogram.org/results/deliverables-and-other-reports/i/27891/1941/ericsson-journey-of-change.

Parnell-Klabo, E., 2006. Introducing lean principles with agile practices at a fortune 500 company. In: Proceedings of AGILE 2006. IEEE Computer Society, Washington, DC, USA, pp. 232–242. doi:10.1109/AGILE.2006.35.

Petersen, K., 2010. An empirical study of lead-times in incremental and agile software development. In: Proceedings of the 2010 International Conference on Software Process. Springer-Verlag, Berlin, Heidelberg, pp. 345–356.

Petersen, K., Wohlin, C., 2010. The effect of moving from a plan-driven to an incremental software development approach with agile practices. Empirical Softw. Eng. 15 (6), 654–693. doi:10.1007/s10664-010-9136-6.

Petersen, K., Wohlin, C., 2011. Measuring the flow in lean software development. Softw.: Pract. Exp. 41 (9), 975–996. doi:10.1002/spe.975.

Reinertsen, D.G., 2009. The Principles of Product Development Flow: Second Generation Lean Product Development. Celeritas Publishing.

Rodríguez, P., Markkula, J., Oivo, M., Turula, K., 2012. Survey on agile and lean usage in Finnish software industry. In: Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering. ACM, New York, NY, USA, pp. 139–148. doi:10.1145/2372251.2372275.

Rönkkö, M., Järvi, A., Mäkelä, M.M., 2012. Measuring and comparing the adoption of software process practices in the software product industry. In: International Conference on Software Process, ICSP 2008 Leipzig, Germany, May 10-11, 2008 Proceedings. Springer, Berlin, Heidelberg, pp. 407–419. doi:10.1007/978-3-540-79588-9_35.

Sjøberg, D.I., Johnsen, A., Solberg, J., 2012. Quantifying the effect of using Kanban versus Scrum: a case study. IEEE Softw. 29, 47–53. http://doi.ieeecomputersociety.org/10.1109/MS.2012.110.

Šmite, D., Moe, N.B., Ågerfalk, P.J., 2010. Agility Across Time and Space: Implementing Agile Methods in Global Software Projects. Springer-Verlag, Berlin, Heidelberg.

Springer. Springer Link. www.link.springer.com/.

Staron, M., Meding, W., 2011. Monitoring bottlenecks in agile and lean software development projects—a method and its industrial use. In: Caivano, D., Oivo, M., Baldassarre, M., Visaggio, G. (Eds.), Product-Focused Software Process Improvement. In: LNCS, Vol. 6759. Springer, Berlin, Heidelberg, pp. 3–16. doi:10.1007/978-3-642-21843-9_3.

The World Café. http://www.theworldcafe.com/method.html.

Wohlin, C., Ruenson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A., 2012. Experimentation in Software Engineering. Springer.

Marta Olszewska (née Pląska) is a postdoctoral researcher at the Distributed Systems Design Laboratory at Åbo Akademi University, in Turku, Finland. Currently she is involved in the Academy of Finland funded project ADVICeS on Adaptive Integrated Formal Design of Safety-Critical Systems. Her research interests focus on establishing and validating metrics and measurements in perspective of how the development processes and practices impact the software quality. She obtained her Ph.D. degree in 2011 with the thesis "On the Impact of Rigorous Approaches on the Quality of Development ". So far he published 18 articles, of which 13 were refereed.

Jeanette Heidenberg D.Sc.(Tech.) is an Innovation and Business Architect at Ericsson Finland. She did her Ph.D. in Software Engineering at Åbo Akademi University in 2011 with a title of "Towards Increased Productivity and Quality in Software Development Using Agile, Lean and Collaborative Approaches". She has over a decade of software industry experience (telecommunications) in the areas of software development and software process improvement deploying agile methods.

Max Weijola holds an M.Sc. in Computer Engineering with a major in Software Engineering from Åbo Akademi University. He currently works as Quality Assurance Manager at Lumi Technologies, a company focusing in audience engagement technology. His research has been mainly focused on agile software development processes and metrics. With a background in both research and industry his goal is to find ways to improve software development processes in a quantifiable way from a combined research and practitioner point of view. He has co-published three articles in various proceedings.

Kirsi Mikkonen is currently responsible of external collaboration and funding within Ericsson R&D Finland. In her role as Change Manager and organizational Coach, she has driven the sustainable self-learning organization culture in both R&D and administrative organizations. She has strong experience in People management, Lean Leadership, Brain based coaching, Group facilitation, Agile and Lean methodology. She has Master of Science in Electrical Engineering from Helsinki University of Technology and Norwegian University of Science and Technology, in 1995.

Ivan Porres PhD (Eng.) is professor in Software Engineering at Åbo Akademi University, in Turku, Finland and the leader of the Software Engineering Laboratory at TUCS, the Turku Centre for Computer Science. He is the principal investigator at Åbo Akademi for the Cloud Software Finland (2009–2013) and N4S (2014–2016) projects at DIGILE, the Finnish Strategic Centre for Science, Technology and Innovation in the ICT. He has received the Ten-Year Most Influential Paper Award at the ACM/IEEE Conference on Model Driven Engineering Languages and Systems in two occasions. He is the author of more than 100 scientific articles in Software Engineering.