Marta Olszewska | Jeanette Heidenberg | Max Weijola | Kirsi Mikkonen | Ivan Porres

Did it actually go this well? A Large-Scale Case Study on an Agile Transformation

Turku Centre *for* Computer Science

TUCS Technical Report
No 1108, May 2014

Did it actually go this well? A Large-Scale Case Study on an Agile Transformation

Marta Olszewska
Åbo Akademi University
Department of Information Technologies
Joukahainengatan 3-5 A, 20520 Åbo, Finland
marta.plaska@abo.fi

Jeanette Heidenberg
Ericsson R&D Center Finland
Hirsalantie 11, 02420 Jorvas, Finland
jeanette.heidenberg@ericsson.com

Max Weijola
Åbo Akademi University
Department of Information Technologies
Joukahainengatan 3-5 A, 20520 Åbo, Finland
max.weijola@abo.fi

Kirsi Mikkonen
Ericsson R&D Center Finland
Hirsalantie 11, 02420 Jorvas, Finland
kirsi.mikkonen@ericsson.com

Ivan Porres
Åbo Akademi University
Department of Information Technologies
Joukahainengatan 3-5 A, 20520 Åbo, Finland
ivan.porres@abo.fi

TUCS Technical Report

No 1108, May 2014

Abstract

Agile software development continues to grow in popularity and is begin 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 and disrupted working routines can get high. Such evidence exists in the form of success stores and case studies, but most of the existing research in this area is qualitative in nature. In this article we provide a quantitative metrics model containing eight rigorously described metrics and their application with a case study evaluating an agile and lean transformation in a large telecommunications organization. Our findings show significant improvement in six of the eight metrics whereas one metric showed deteriorated results.

TUCS Laboratory
Software Engineering Laboratory

1 Introduction

The IT world of today remains highly competitive and value oriented. Due to constant business and technological changes in requirements or in the environment, companies active in this area strive to be flexible and adaptive to change. Therefore, agile and lean software development methods are far from mere buzz-words popular only in certain small, innovative organizations. Rather, they are gaining popularity among companies of various sizes and domains [31, 11]. Agility is in itself a desired characteristic with over a decade of successful adoption both as a complete development process or customized to a pre-existing organizational process. One popular customization is the combination of the lean thinking concept with agile software development resulting in a powerful combination.

It takes time, effort and resources to transform the way of working and thinking in a large-scale organization. Therefore, it needs to be quite likely that such change will be beneficial for the company and the employees before starting such transformation. Evidence that this is the case can be gathered, e.g., from empirical investigations into existing cases. It is essential to provide such information not only for the sake of existing organizations, but also for the purpose of enriching the software engineering body of knowledge.

The impact and benefits of agile and lean software deployment have been investigated previously and reported in [34, 1, 23, 26, 29]. Most of these studies are, however, of a qualitative nature. There is a need for further empirical study[14, 11], but also for quantitative results [13].

In this work we present a quantitative comparative case study in the context of a large-scale telecommunication company. We are exploring an organization before and after agile and lean transformation. We investigate the impact of the change on the software development process in order to provide information on the effect of such a change. In particular, we use metrics and measurements that are feasible to analyze both plan driven development and agile and lean development.

Based on the state of the art, our research goal, as well as the context of our research, we formulate the main question to be answered in this study as:

RQ: *What are the measurable impacts of the changes in the development process before and after an agile and lean transformation?*

We utilize the metrics model presented in our previous work [18], where we proposed a set of metrics for evaluation of improvement of the development process. In this paper we describe in more detail the earlier established metrics and extend them by formulating them in a neutral, structured and formalized manner (the change itself is assessed – not the improvement) and by further developing the model based on empirical data. The key idea behind the metrics and measurements in this paper is 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.

Experience and continuous learning are central tenets in both the agile and lean communities. With our results we contribute to this idea by building the body of knowledge in the agile and lean domain as well as enabling further empirical investigations by providing a re-usable metrics model.

Agile methods typically use time-based measures, which intentionally have no standard definition, as the goal is team autonomy and self-organization rather than consistency across teams or projects. In this work, we provide a comprehensive description of metrics, so that their application is transparent and their use by others in the future is reliable and straightforward.

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 to a case study on a large organization from the telecommunication domain. We perform verification and 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 case study. We examine our results both on a general level (metrics definitions), as well as study their usefulness (case study). We examine our results on a generic level, i.e., the feasibility of the proposed metrics for the purpose of the case study.

Our results are presented in such a way that organizations of similar size or organizational context could use them as a reference for answering their concerns about carrying out an agile and lean transformation.

The remainder of this paper is structured as follows. Section 2 introduces some relevant terminology and related work, followed by a description of topics related to 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, Section 6 contains a discussion on validity concerns regarding the metrics. The main results of the case study are presented in Section 7. Discussion and conclusions are given in Sections 8 and 9 respectively, thus finalizing the paper.

2 Background and Related Work

Since the terms *metrics*, *measurement* and *large-scale* have been used in different ways in literature and the terms can have varying meaning in different contexts, we want to provide the terminology that we rely on in this work in section 2.1. Related empirical investigations are presented in Section 2.2,

where we focus on findings where quantitative data is reported.

2.1 Terminology

Metrics and measurements are measurement concepts that provide awareness on certain aspects of quality. Thus, in order to measure given quality attributes, we want their properties to be expressed in terms of those measurements.

In this work we define *metric* in accordance to the IEEE Standard glossary of software engineering terminology as a quantitative measure of the degree to which an item possesses a given quality attribute [37].

A *measurement* is defined in accordance with Fenton & Pfleeger as a number or symbol assigned to an entity by the measurement mapping in order to characterize an attribute [15]. Thus, we understand a measurement as an application of a metric to a specific entity.

This paper focuses on a *large-scale* agile and lean setting. However, what constitutes large-scale is still being discussed in [11], where Dingsøyr and Moe present a number of factors that have been suggested as definition for large-scale: (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 [11, 10]. In our work we agree on defining large-scale by number of teams. However, in contrast to Dingsøyr and Moe, we consider the minimum number of teams to be at least three in order to constitute a large-scale agile development setting.

2.2 Related Empirical Investigations

When examining the existing literature with regard to empirical investigations related to agile and lean transformations, many studies are to be found. However, most of the related work is of a qualitative nature. Our focus is on reporting quantitative results.

Talby & Dubinsky report findings related to software project governance [36] with both qualitative and quantitative data. However, their focus is on single sprint iterations in a project using only extreme programming (XP) as development processes. Swaminathan & Jain reports [35] quantitative results related to introducing lean principles into agile software development. Again the study examines only agile development methodologies with one development team. A similar report is presented by Sjøberg, Johnsen and Solberg [33] where indeed quantitative results are presented, although comparing two agile methods against each other (Kanban vs Scrum). Although

we find these studies beneficial and interesting, we focus on plan driven versus agile development methodologies.

One report that shares our focus on the plan-driven "old" versus agile "new" development methods is presented by Li, Moe & Dybå [24]. Li et al. reports a longitudinal case study with focus on software quality, comparing a plan-driven development setting with Scrum. In contrast to the Li, Moe & Dybå study, in our work we aim to focus on a broader set of attributes than purely software quality.

3 Investigation Strategy

In this section we describe our case study by first explaining the research method (Section 3.1), followed by the description of the context of our investigation (Section 3.2). We continue with the portrayal of the data collection process (Section 3.3). Finally, we post the research questions and describe the context of the data collection in Section 3.4. The details and specific challenges regarding data collection are to be found in the section describing the case study (Section 7.1)

3.1 Research Method

We chose *case study* as our research method, mainly for two reasons: (i) case study is a flexible investigation strategy that allows us to observe and understand the effects of a change, (ii) practice is often ahead of research [11]. We perform our investigation following the guidelines given by Runeson and Höst [32].

Our study has an exploratory objective, meaning that we want to investigate the impact of changing the development process on the organization and possibly obtain some new insights. In the long run, there is also an improvement aspect present in our research work, since the metrics we use now for the comparison can be used for providing transparency in the development in the future. Our case study is considered embedded [32], as we investigate two development processes in the same organization, each for a certain period of time. The investigation is classified as *sister projects* [15], since the environment in the research is the same, only the development process differs.

3.2 Research Context and Case Description

The key drivers for this research were identified in the *Cloud Software Finland* [6] 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.

In this paper we present the results of iterative collaboration with *Ericsson R&D Center Finland* – a large-scale software development telecommunication company that has completed an agile and lean transformation. There has already been some research concentrating on qualitative results of the agile and lean transformation [19]. However, a need for providing quantitative measurements was identified by the case company, and further motivated by the gap in the state of current research [13]. Therefore, in this work we pursued the aspect of quantitatively measuring the transformation from plan driven to an agile and lean development process in order to complement current qualitative research.

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 telecommunication product is roughly 10 years old and highly complex, consisting of RoseRT, C++ and Java code [22]. 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 towards 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 [22]. 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.[9], 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 [22]. The number of agile teams changed slightly according to needs during the transformation. However, the total number of teams was usually more than 20 and hence fulfills the definition of large-scale according to [11, 10] (see section 2.1).

As the transformation was inspired by success stories, it was so complex that it could not be justified with regular business case procedures. Therefore, both qualitative and quantitative studies were in demand in the case company.

3.3 Units of Analysis

Within the research context and case description above, we are interested in how the change in the development methodology impacted the development process and artifacts involved. We want to measure the impact neutrally, not with a desired outcome in mind (e.g. improvement). The metrics model can, in this perspective, simultaneously be seen as a quality model, since the attributes defined in the metrics model are implicitly related to quality attributes [20].

The data was 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 [38], 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 investigation 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 of investigating the before and after transformation artifacts. This type of investigation is objective and reflects the actual timeline 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 the investigation (some files contained more restricted dates). Therefore, after further discussions with experts from the organization, we chose the most representative time intervals, i.e. 2008-2009 for the old way of working, 2010 for the transformation period and 2011-2012 for the new way of working.

3.4 Research Questions

We use the Basili et al. *goal template* [3, 38] for defining the purpose of our experimentation. The goal of our study is to:

---

analyze the plan driven versus agile and lean development
for the purpose of comparison
with the respect to the impacts of the changes in the development process
from the point of view of the organization and the researchers
in the context of a large-scale telecommunication organization.

In the light of this goal, we continue to discuss our main research question, mentioned in the introduction, i.e.

RQ1: *What are the measurable impacts of the changes in the development process before and after an agile and lean transformation?*

Since the question is quite generic, we refine it by specifying what we mean by 'the development process'. The term includes 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.

Considering these identified issues, we propose the following questions to be answered by the comparison of the periods before and after the transformation from the point of view of the organization and the researchers:

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?*

These four questions are further discussed in the context of the metrics model in Sections 4 & 5.

4 Metrics Model

The quality model we present in this paper is a refined version of the one that was introduced in our previous work [18]. In general, the model was

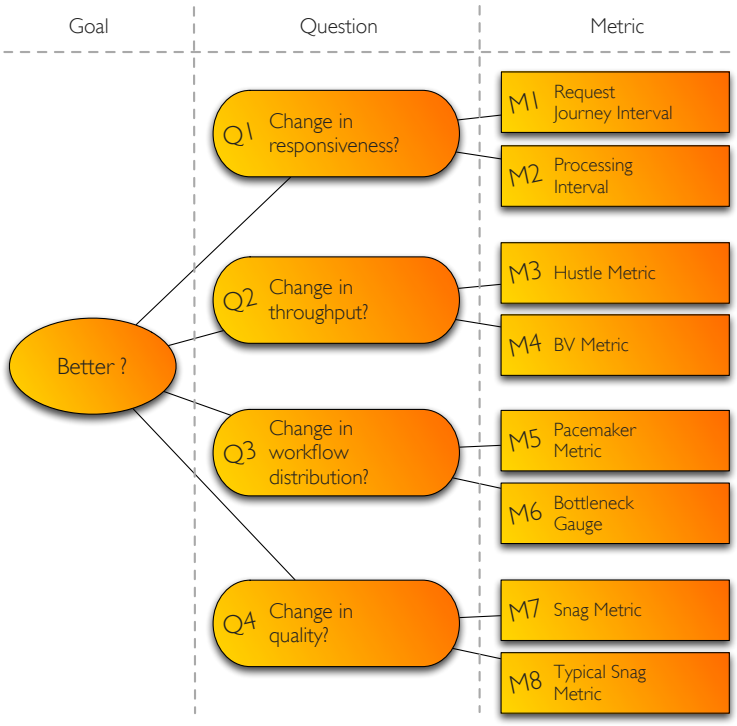


Figure 1: Visualization of the metrics model

developed to quantitatively compare a software development organization before and after an agile and lean transformation. Moreover, it should serve as a complement to qualitative studies, i.e. interviews and surveys, performed so far. The model components were iteratively developed with a number of industry partners within the *Cloud Software Finland* research project. The metrics are elicited with the use of the *Goal Question Metric* approach [4]. The resulting metric model is presented in Figure 1. It consists of four questions (Section 3.4), for each of these questions there are two metrics established (subsections 1-4 in Section 5). Depending on the availability of data and rigour 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.

Our contribution to the model in this paper is the refinement of the established metrics. We concentrated on defining and detailing the metrics in such a manner, that they are feasible for providing measurements for comparison purposes, while remaining valid for both traditional as well as agile and lean development processes.

Since we wanted our metrics to be practical, usage of existing data sources [4] was one of the key aspects which we prioritized highly when constructing

the metrics. It 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. Metrics themselves are of a minor value if their calculation is not possible (e.g. due to the lack of data). Therefore, the data availability played a vital role in the choice of metrics for the metrics model. We rely on the rich dataset collected from the organization in order to obtain meaningful results. The data collection process is mostly automatic and objective, i.e. no subjective judgment is required to obtain the data.

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 3.4. Metrics should provide means and facilitate continuous self-assessment and improvement. Moreover, they need to provide transparency to project status evaluation, as well as act informatively with respect to e.g. customers or 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 [16]. 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.

5 Description of Metrics

We concentrate on software quality metrics and measurements (see Section 2.1) 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 et al. [21] with property-based software engineering measurement given by Briand et al. [5]. In consequence, we provide a detailed, but also concise description of the purpose, scope and attributes of measurements, as well as the measurement scale, variability and possible relations between metrics and attributes. Finally, we conclude each metric description with some mathematical properties and dimensional analysis, if there are any. The discussion regarding side effect of using measurement instruments, measurement errors and validity concerns is presented separately in Section 6.

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 [30] 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 [28].

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) [7].

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.

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

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.

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))

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 [28]. 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.

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

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 $T_{Ship} > T_{Impl}$.

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.

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 Andersson [2] and Petersen & Wohlin [27], 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 [17]) can be delivered in relation to a certain work effort. The proposed metric could be computed as the ratio of test points, as described by Dubinsky et al. in [12, 17], 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 sellable licenses per money spent. Sellable licenses Δ[1] are defined as single or grouped features, which are possible to sell as a certain functionality to a customer. This metric (similarly to Metric 1) also supports 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 how much money or work effort needs to be spent on implementing and delivering 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

[1] Δ Jeanette: Ivan suggested changing the name from sellable license to sellable feature pack OR sellable license type. The motivation was that sellable license can be interpreted as 10 licenses sold for one type (e.g. windows vista), while the sellable type is only one product. Sellable license types would be the portfolio, and not account for the actual number of sold licenses. Do you agree on changing the terminology to feature packs/ license types?

is, which can be used for assessment of organizational improvement with respect to the organizational performance.

Attribute Throughput.

Natural scale (Attribute) Sellable licenses, defined as a feature, a product or a service for which the company can charge the customer - absolute scale (simple count), money spent - ratio scale.

Variability (Attribute) The number of the sellable licenses 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 $\frac{Sellable\ licenses}{Money\ spent}$, where sellable licenses and money spent are natural numbers, both greater than zero.

Natural scale (Metric) Ratio scale dictated by the money spent element, since it is represented by a weaker scale than absolute scale (sellable licenses). There is a "zero point", which is when there are no sellable licenses 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 sellable licenses 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 sellable licenses), disjoint attribute additivity (two disjoint licenses 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 sellable licenses 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 / Time period)

Purpose Business value is measured as more frequent major releases [8, 16] 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 users needs 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 [16]. 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.

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

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).

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 towards 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.

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 [12, 17], 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.

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

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.

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)

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 [27] 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).

Natural scale (Attribute) Ratio scale.

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.

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.

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).

Natural scale (Attribute) Absolute scale (one measurement mapping – a simple count).

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.).

Natural scale (Metric) Absolute scale, it is a number of occurrences of the external trouble reports. Can be obtained from a specific timeframe (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)

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 complicated or that the code base is more difficult to maintain. Both of these are indications that the quality of the product has deteriorated. Furthermore, 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).

Natural scale (Attribute) Absolute scale (one measurement mapping – a simple count of 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".

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.

The goal of the thorough description of metrics is to assist organizations and developers in the assessment of the usability of the presented metrics in the context of their application. In order to be able to make an informed decision about including these metrics in any development setting, we follow our discussion on metrics with their validation in Section 6.

6 Validation of Metrics

In this section we first discuss the side effects of using measurement instruments, followed by the measurement errors and validity concerns on a general level for all the metrics. This analysis is then completed by specific validity concerns that exist for certain metrics only. Finally in Subsection 6.2, we enhance and support the validity description by applying the validation criteria given by Meneely et al. [25].

6.1 Validity Concerns of Metrics

The side effects of using measurement instruments to the presented metrics can be discussed only if a measurement itself has an negative impact on the developers behavior or the way the development progresses, i.e. distorts the employee's behavior and thus provides less value to the organization. In our case there are no foreseeable side effects, since the measurements are not used for judgmental purposes, but rather as information radiators. Moreover, it is not possible to tamper with the timestamps (metrics 1, 2, 5, 6 and 8) and time periods (metric 4 and 5), as they are collected automatically. Furthermore, simple counts of sellable licenses, major releases, number of days or trouble reports (metric 3, 4, 5 and 7, respectively) are rather straightforward and therefore not anticipated to cause any distortion.

The presented metrics are 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.

One should keep in mind that for a certain measurement application, many properties will be specific to the environment and the organization where the metrics have been applied. This is where domain knowledge supports the measurement process to create viable empirical evaluation.

A practical implication of using the metrics proposed in this work is gaining a deeper understanding of the software development processes dynamics (in particular the observable characteristics between different development approaches). Another implication is the identification and measurement of improvement areas in the development process. In the following subsection we employ the *Goal Driven Philosophy* and discuss the validity concerns according to the framework presented by Meneely et al. [25], as we are in principle interested in the usefulness of metrics.

6.2 Applying Validity Framework to Metrics

We follow the validity scheme described by Meneely et al. [25], where the authors listed 47 validity criteria and thoroughly described their semantics. In our work we use a subset of 21 criteria (denoted in *italics*) that are applicable for our setting. These are appropriate for the metrics only, rather than the case study as a whole. The validity of a case study is discussed later (Section 7.3).

The priority for our metrics is their *applicability*, instead of them being theoretically correct. We do, however, provide some *mathematical properties* of the proposed metrics without mathematically proving these properties.

Providing the measurement scale and hinting some mathematical properties serves to indicate how the metric can be used, i.e. the possible statistical operations (as well as statistical tests) that could be performed on the obtained measurements. The *scale validity* is preserved, meaning that an explicit and appropriate measurement scale is defined so that all meaningful transformations of the metric are admissible. Moreover, the *unit validity* is taken under consideration, as for each metric we propose suitable means of measuring the attribute in question. We also shortly discuss the *monotonicity* of each metric and check whether the development of a single component or feature is no more complex or takes longer to develop than the entire program.

Our metrics descriptions are deliberately quite thorough. Our goal is to provide a *clear definition* for each of the metrics, so that other researchers would not be misled or draw wrong conclusions about them. Additionally, the *internal validity*, meaning how well and correct do the metrics measure certain attributes, is supported, since it is related exclusively to the metrics themselves and not the external quality factors. Our metrics are conceptually interrelated (see the quality metrics model, Figure 1) and facilitate the interpretation of the measurement results, also with visual means. Therefore the *internal consistency* is considered.

We firstly create opportunity for potential users of metrics to check the feasibility of the metrics for their purpose (*transferability* of metrics). Secondly, we aim for the metrics to be accurately applied in other projects

(*repeatability* of metrics). Our ambition is to demonstrate that the metrics are repeatable, i.e. to show that they are empirically valid, in two settings: plan driven as well as agile and lean software development in a given organization. We are aware that we are not able to declare universal repeatability or generalizability of the metrics yet, as the metrics have not been tested in a sufficient number of other settings or projects. Due to the quantitative nature of our metrics, there is little space left for bias or random errors related to human perception. Thus *accuracy* and *reliability* are sustained. Furthermore, our metrics exhibit *non-exploitability* characteristic, since the developers or teams cannot manipulate a metric to acquire the results they wish nor impact the attribute being measured.

The metrics presented have *construct validity*, as the gathering of measurements for these metrics corresponds to the definition of the targeted attribute. Moreover, the operational definition produces data related to an abstract concept. The metrics are mostly based on time measurements, as well as simple counts, all of which are automatically collected. Their construct validity is rather straightforward. However the functionality and business value metrics are compound and their meaning and computations are organization-specific. The construct validity cannot be therefore generally confirmed for these two cases – it can only be confirmed in the perspective of their application to our case study.

The goal of these metrics is to aid the understanding of software quality, i.e. give *constructive* feedback on the quality and aspects related to quality, e.g. development process oriented qualities. The metrics are *actionable* as they enable a software development team or a software developer to make empirically informed and timely decisions based on the software product's status during development. The metrics reflect certain properties of software and by that serve as guidelines or recommendations for development.

Presented metrics are *relevant* and *usable*, i.e. they are generally defined at first, then tailored and finally cost-effectively implemented by a given organization. We established and *empirically* validated the metrics in the domain of a large telecommunication company. Our experimentation corroborated the intended measurements of given metrics, as well as the relationship between the metrics and respective software quality factors (*external validity*). The metrics have potential to be transferable to other organizations and domains, as well as to be feasibly included in the quality assurance practices or development process.

We support the *economic productivity*, as using our metrics implicitly quantifies a relationship between cost and benefit. One of the goals of application of metrics is for them to serve as catalysts in the development process by reinforcing the development with quantitative data. Ultimately, an implicit goal of applying a metric should be saving resources (such as development time and money) and provide other cross-organization benefits

(transparency of information, possibility of self-organization, etc.). Collecting measurements for given metrics, as well as achieving viable measurements should be and is economically feasible, since metrics are well described and tailored to the needs of certain development type and its environment, and the data collection is automated.

This discussion on validity of metrics was necessary in order to show that we use them in our case study in a reasonable manner. Being aware of the limitations of our metrics is crucial to obtain meaningful results. Furthermore, it is important for other organizations and researchers to understand the restrictions of our metrics in case they would be utilized in other context. Finally, it gives a perspective on the purpose of our metrics and some of their characteristics on a more general level.

7 Case Study Results

In order to fulfill the goal of our investigation and answer the four research questions, we apply the proposed metrics to our case study. Therefore, in the following subsections we describe the process of data collection and analyze the gathered data. Furthermore, we discuss the validity issues concerning large, complex and real-life case study.

7.1 Data Collection

In our work we rely on multiple data sources, i.e. the 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 Figure 2. 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).

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. aggre-

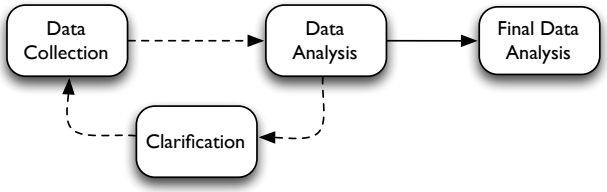


Figure 2: Visualization of the data collection process

gated 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 case study.

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 [12] delivered per time unit, but was altered to number of sellable licenses per money spent.

7.2 Analysis

The data analysis was done in several iterations, in parallel with data collection as illustrated in Figure 2. First, the basic and self-explanatory data was collected, studied and the obtained results were presented to the experts. Afterwards the more complex (and pre-processed) data was gathered, analyzed, and discussed together with experts. 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 dataset we collected is of considerable size, however due to confidentiality reasons we cannot provide a detailed description here. We may however reveal that e.g. only 5% of trouble reports logs were issued by customers, and there were twice as much of customer service requests (mostly related to the extensions of the contemporary development) than trouble reports. The aforementioned data is collected for the period from the year 2006 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 30 - 32, 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.

The analysis for Metric 1 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 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. 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 $\frac{Sellable\ licenses}{Money\ spent}$ 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 sellable licenses per money spent is 483 % times higher in the new WoW than on the old WoW.

The data for Metric 4 was retrieved in the same workshop as for the Metric 3, 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.

Metric 5 was analyzed from a single file of raw data extracted from a version control system that was made available to the researchers. 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

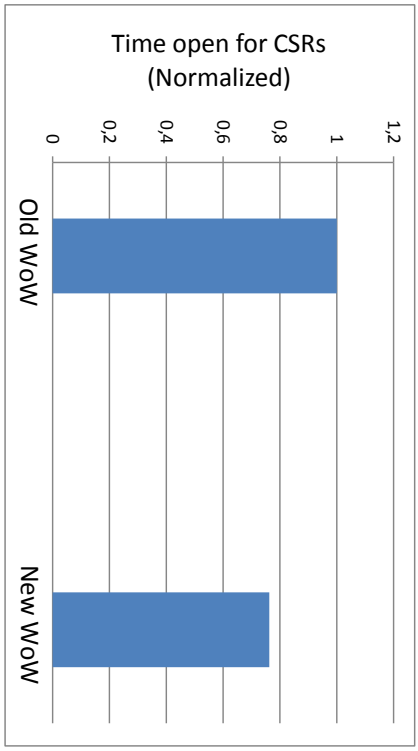
days between commits decreased from 12 down to 4. For the remaining days (1 to 4 days between commits), the occurrences decreased on a weighted average by roughly 38 %.

Metric 6 was the most challenging of all metrics. In our case study setting, we were not able to provide an analysis with regards to the data set obtained in our research. When developing the metric, we were inspired by the work of Petersen & Wohlin [27]. For the old WoW we were able to obtain data for metric 6 in accordance to the original description of the flow metric [27]. In the new WoW we were not able to retrieve data that would satisfy our needs for the original flow metric description. After discussion between the researchers and company representatives, the current description of flow metric was developed (based on HandoverTimes) and a data set was obtained. However, after careful initial analysis and consultation with the case company, it turned out that the two datasets for the old and new WoW were not comparable. A method for converting the old WoW data to correspond to the new WoW data was proposed but it was considered too expensive to perform and would violate the properties of the metrics model to be practical and easy to obtain. A further discussion regarding the challenges of this metric is presented in the Discussion in section 8.

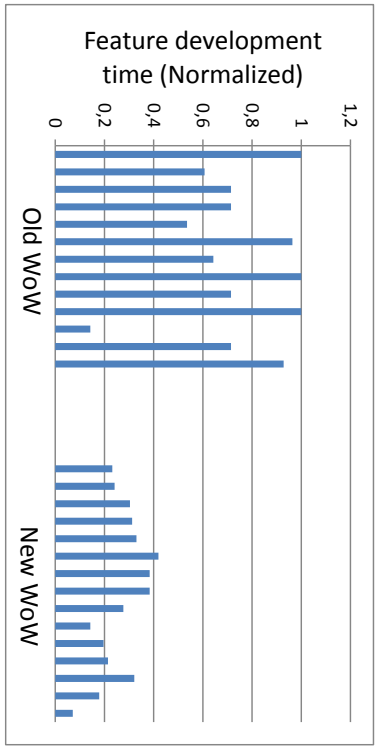
The analysis for Metric 7 was based on a single data file of raw data, which was provided by the case organization. Dataset reduction was performed to conform with releases pertaining to the old and new WoW, respectively. The results show an 188 % increase in Trouble Reports. As this metric showed a degradation in performance, the decision was taken to also add data from the transformation period to evaluate the trend for trouble reports.

Metric 8 was analyzed based on the same data file as Metric 7 with the same dataset reduction. The results for the average time that external Trouble Reports remain open show a decrease of approximately 31 %.

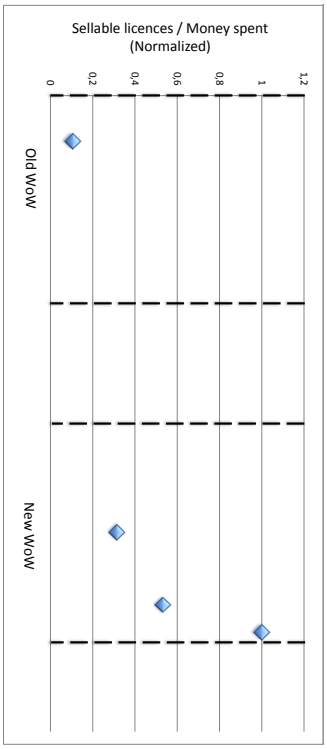
In this subsection the analysis process was described and the results presented. To ensure a fair discussion of the results in Section 8. we discuss threats to validity in the following subsection.



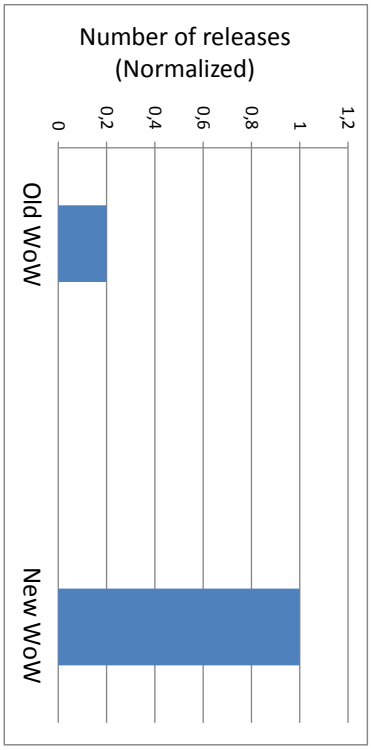
(a) Metric 1 - Request Journey Interval



(b) Metric 2 - Processing Interval



(c) Metric 3 - Hustle Metric



(d) Metric 4 - BV Metric

7.3 Validity Issues

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 paper we already reviewed the validity threats with respect to metrics (Section 6), thus providing the analysis on a generic and abstract level. Now, we analyze the validity of our results, i.e. the application of the proposed metrics in a case study.

We follow the validity discussion scheme proposed for software engineering domain by Wohlin et al. [38]. The analysis distinguishes four different types of validity, dealing with the issues of causality (internal validity), generalization of findings (external validity), relation between theory and practice (construct validity) and relationship between cause and effect (conclusion validity).

7.3.1 Internal Validity

Discussion on internal validity studies the relationships between causes and effects. In our case we want to ensure that the agile and lean transformation caused the observed changes (the effect). We examine that the changes were not resulting from factors of which we had not planned or measured.

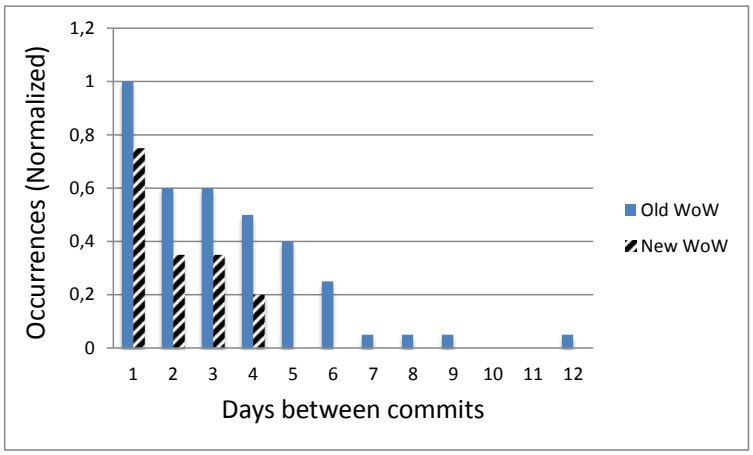
While the selection of subjects had no impact in our investigation (motivation and suitability of persons involved was appropriate), the non-human related factors were affecting the complexity of the case study, e.g. impacting the data collection process.

The instrumentation, although bringing objectivity, triangulation and automation to our work, entailed some difficulties due to having multiple tools providing different format of data. These low level challenges had to be tackled when working on feasibility of datasets (dataset reduction).

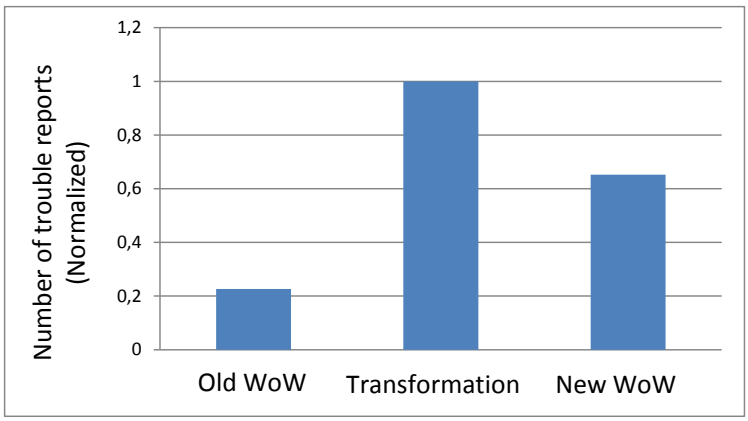
7.3.2 External Validity

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.

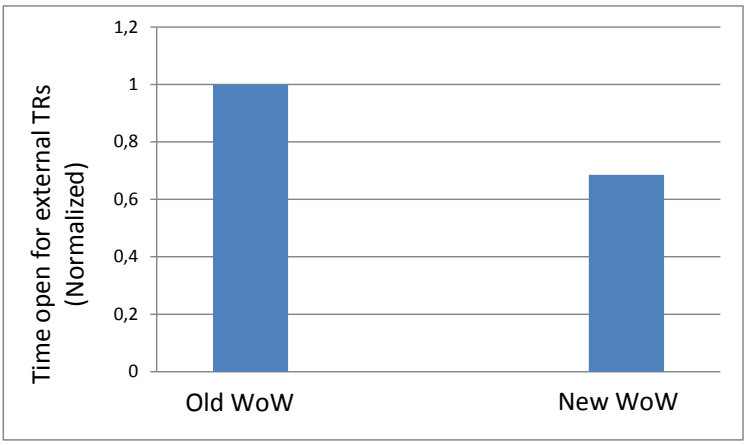
The type of chosen investigation strategy (case study) makes it difficult to generalize the results. The presented outcomes can be to some degree generalizable in contexts similar to ours, i.e. large-scale and customer driven organizations, which are either planning, executing or are after the agile and lean transformation. Also organizations with iterative and incremental development processes may find our results applicable to their setting.



(a) Metric 5 - Pacemaker Metric



(b) Metric 7 - Snag Metric



(c) Metric 8 - Typical Snag Metric

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

Table 1: Summary of the results obtained from the collected data sets in the case study.

We kept the environment as realistic as possible, therefore the application of our metrics model to other industrial practices has potential. Our aim was to not only show a result for a certain company, but also enable the utilization of our model by companies of a comparable size, development process and application domain. The findings are envisaged to provide valuable insight to practitioners and researchers, who are interested in applying the metrics model themselves.

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. Yet, the measurement of a change in throughput should remain as one of the factors monitoring of how much value the organization delivers.

7.3.3 Construct Validity

When discussing construct validity we are concerned if we can generalize the constructs, i.e. 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.

The results of application of metrics to the case are itself interesting. Since we investigate the problem of a "change" from many perspectives, i.e. having a number of metrics, 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 case study with respect to the available data.

7.3.4 Conclusion Validity (Reliability)

When collecting data there often rises a challenge of the data collection and the completeness of the obtained dataset. Obviously, the quality of the data is reflected in the final outcome of the investigations. In most cases large datasets are beneficial for the investigations (large sample size), but at the same time may also obscure the problems with the quality of data. In our case study we tackle such large dataset, but we do not use all the collected data due to the incompleteness of records, e.g. lacking the start or end dates, not completed development or features. Therefore, a dataset reduction was necessary. Some data points were removed since they did not have a specified end date, but only a release date or release number (the release date might be significantly later than the feature completion date). The lack of a few data points could in most cases be compensated by results of other metrics (functionality could be balanced by the BV metric).

One important topic to be mentioned is that we are proceeding with our case study in a time-frame manner, meaning that some of the older data (originating before the explicitly defined investigation periods) were "cut-off" and it is not possible to state what impact this data could have had on the outcome. However, also in this case we discussed the problem with the experts from the organization and as a result the most feasible periods of time were chosen.

In our work we are comparing the development processes before and after the transformation, however we are not interested in statistical significance. We are more concerned with the careful data collection and definition of metrics, in order to obtain a meaningful measurements in our case study. We place emphasis on having the same outcome of a measurement, if the measurement is done many times.

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.

.

We regard our metrics model as generic enough to be tailored for the specific needs of organizations. However in this paper we propose concrete metrics (Section 5) that were applied in the studied organization. Most of the data is collected automatically and the metrics are computed in the same way. We are not claiming that none of the metrics can be manipulated, but when having explicitly defined metrics and limiting the human factor in the measurement process, this risk is very much reduced.

8 Discussion

So far, measurements of agile development process and especially the organizational changes leading to establishing this process in large-scale organizations have been neglected in research[13]. In this paper we focused on providing metrics and quantitative measurements of the transformation from plan driven to agile and lean development process performed in a large telecommunication organization.

The choice and definition of metrics was dictated by 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.

There exists several types of metrics that are well known, discussed in literature and used in practice, but were not included in this work. These are, amongst other, productivity metrics, such as lines of code per person, capacity utilization and code churn [33]. We anticipated that metrics such as these can be directly targeting personnel abilities, and thus be destructive and demotivating (and at the same time hindering the agile principles). The metrics we established were process oriented, related to both internal and external attributes, i.e. functionality and execution related.

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), which we consider not as a deterioration of quality; rather, we reason that it is because of substantially more functionality being implemented (M4) and delivered in shorter time (M2). When looking at the data from the transformation period, a decreasing trend can however be identified. 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.

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 ways of working 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 from this case study, M6 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 33, 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 (revenue or user experience).

9 Conclusions

This work is a result of our collaboration with our industrial partner (Ericsson) on providing quantitative data and analysis to measure impacts of the changes in the development process after the agile and lean transformation that took place in the organization in 2010. We defined metrics in a

systematic and thorough manner based on work from Kaner et al. [21] and Briand et al. [5] through numerous iterations on the initial proposal of metrics presented in our previous work [18]. Moreover, we presented a complete case study from the telecommunications domain with detailed analysis of collected and processed data. Finally, we provided validation of our approach on two levels: separately for metrics and for the case study.

Often agile and lean software development methods are motivated by success stories and qualitative studies. Our goal with this work was to contrast the existing qualitative studies with a quantitative one. The task was not straightforward and required many iterations between the researchers and the case company. Examples of factors that contributed to complexity were: change in software supporting development, challenges in data retrieval from older projects and mapping of metrics between old and new way of working.

Besides being used for comparison purposes, the actionable goal of our proposed metrics was to be motivational, i.e. to demonstrate the current state of development and encourage continuous improvement. Additionally the metrics are informational, enhancing the overall understanding of the current state of development.

Six out of the eight metrics showed a clear improvement due to the agile and lean transformation, where only one displayed well justified deterioration. The advantage of our metrics in the comparison is their objectivity and repeatability, which is achievable due to the highly automated data collection process.

References

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

[2] David Andersson. *Agile management for software engineering : applying the theory of constraints for business results*. Pearson Education inc., 2004.

[3] V. R. Basili and H. D. Rombach. The tame project: towards improvement-oriented software environments. *IEEE Trans. Softw. Eng.*, 14(6):758–773, June 1988.

[4] Victor R Basili, Gianluigi Caldiera, and H Dieter Rombach. The Goal Question Metric Approach. In *Encyclopedia of Software Engineering*, pages 646–661. Wiley, 1994.

[5] Lionel C Briand, Sandro Morasca, and Victor R Basili. Property-Based Software Engineering Measurement. *IEEE Transactions on Software Engineering*, 22(1), 1996.

[6] Cloud Software Finland. Cloud software finland. 2013.

[7] Alistair Cockburn. What engineering has in common with manufacturing and why it matters - ac, September 2006.

[8] Mike Cohn. *Succeeding with Agile: Software Development Using Scrum*. Addison-Wesley Professional, 2009. ISBN 978-0321579362.

[9] Pete Deemer, Gabrielle Benefield, Craig Larman, and Bas Vodde. The scrum primer. Technical report, 2010.

[10] Torgeir Dingsøyr, Tor Erlend Fægri, and Juha Itkonen. What is Large in Large-Scale? A taxonomy of Scaling in Agile Software Development. *Work in progress*, 2013.

[11] Torgeir Dingsøyr and Nils Brede Moe. Research challenges in large-scale agile software development. *SIGSOFT Softw. Eng. Notes*, 38(5):38–39, August 2013.

[12] Yael Dubinsky, David Talby, Orit Hazzan, and Arie Keren. Agile Metrics at the Israeli Air Force. *Development*, 2005.

[13] Tore Dybåand Torgeir Dingsø yr. Empirical studies of agile software development: A systematic review. *Information and Software Technology*, 50(9-10):833–859, August 2008.

[14] Christof Ebert, Pekka Abrahamsson, and Nilay V. Oza. Lean software development. *IEEE Software*, 29(5):22–25, 2012.

[15] Norman E. Fenton and Shari Lawrence Pfleeger. *Software Metrics: A Rigorous and Practical Approach*. PWS Publishing Co., Boston, MA, USA, 2nd edition, 1998.

[16] Deborah Hartmann and Robin Dymond. Appropriate Agile Measurement: Using Metrics and Diagnostics to Deliver Business Value. In *AGILE 2006 Conference (Agile'06)*, pages 6 pp.–134. IEEE Computer Society, 2006.

[17] Orit Hazzan and Yael Dubinsky. *Agile software engineering*. Undergraduate topics in computer science. Springer, Berlin, 2008. ISBN: 978-1-84800-199-2.

[18] Jeanette Heidenberg, Max Weijola, Kirsi Mikkonen, and Ivan Porres. A metrics model to measure the impact of an agile transformation in large software development organizations. In Hubert Baumeister and Barbara Weber, editors, *Agile Processes in Software Engineering and Extreme Programming*, volume 149 of *Lecture Notes in Business Information Processing*, pages 165–179. Springer Berlin Heidelberg, 2013.

[19] Ville Heikkilä, Maria Paasivaara, Casper Lassenius, and Christian Engblom. 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 Engneering and Extreme Programming*, LNBIP, pages 195–209, Berlin, Heidelberg, 2013. Springer-Verlag.

[20] ISO/IEC. ISO/IEC 25010 - Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models. Technical report, ISO/IEC, 2010.

[21] Cem Kaner and Walter P Bond. Software Engineering Metrics : What Do They Measure and How Do We Know ? *Direct 2004*, 8:1–12, 2004.

[22] Kirsi Mikkonen et al. 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`, 2012.

[23] D. Leffingwell. *Scaling Software Agility: Best Practices for Large Enterprises*. The Agile Software Development Series. Prentice Hall, 2007.

[24] Jingyue Li, Nils B Moe, and Tore Dybå. Transition from a plan-driven process to Scrum. In *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement - ESEM '10*, page 1, New York, New York, USA, 2010. ACM Press.

[25] Andrew Meneely, Ben Smith, and Laurie Williams. Validating software metrics: A spectrum of philosophies. *ACM Trans. Softw. Eng. Methodol.*, 21(4):24:1–24:28, February 2013.

[26] Emma Parnell-Klabo. Introducing lean principles with agile practices at a fortune 500 company. In *Proceedings of AGILE 2006*, pages 232–242, Washington, DC, USA, 2006. IEEE Computer Society.

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

[28] Kai Petersen. An empirical study of lead-times in incremental and agile software development. In *Proceedings of the 2010 international conference on software process*, ICSP'10, pages 345–356, Berlin, Heidelberg, 2010. Springer-Verlag.

[29] Kai Petersen and Claes Wohlin. The effect of moving from a plan-driven to an incremental software development approach with agile practices. *Empirical Softw. Engg.*, 15(6):654–693, December 2010.

[30] Donald G. Reinertsen. *The Principles Of Product Development Flow: Second Generation Lean Product Development*. Celeritas Publishing, 2009.

[31] Pilar Rodríguez, Jouni Markkula, Markku Oivo, and Kimmo Turula. Survey on agile and lean usage in finnish software industry. In *Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement*, ESEM '12, page 139, New York, USA, 2012. ACM Press.

[32] Per Runeson and Martin Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical Softw. Engg.*, 14(2):131–164, apr 2009.

[33] Dag I.K. Sjøberg, Anders Johnsen, and Jorgen Solberg. Quantifying the effect of using kanban versus scrum: A case study. *IEEE Software*, 29:47–53, 2012.

[34] D. Šmite, N.B. Moe, and P.J. Ågerfalk. *Agility Across Time and Space: Implementing Agile Methods in Global Software Projects*. Springer, 2010.

[35] Balachander Swaminathan and Karuna Jain. Implementing the Lean Concepts of Continuous Improvement and Flow on an Agile Software Development Project: An Industrial Case Study. In *2012 Agile India*, pages 10–19. IEEE, February 2012.

[36] David Talby and Yael Dubinsky. Governance of an agile software project. In *2009 ICSE Workshop on Software Development Governance*, pages 40–45. IEEE, May 2009.

[37] The Institute of Electrical and Eletronics Engineers. Ieee standard glossary of software engineering terminology. IEEE Standard, 1990.

[38] Claes Wohlin, Per Ruenson, Martin Hst, Magnus C. Ohlsson, Bjrn Regnell, and Anders Wesslén. *Experimentation in Software Engineering*. Springer, 2012.

Turku
Centre *for*
Computer
Science

Joukahaisenkatu 3-5 A, 20520 TURKU, Finland | www.tucs.fi

University of Turku
Faculty of Mathematics and Natural Sciences
- Department of Information Technology
- Department of Mathematics

Turku School of Economics
- Institute of Information Systems Sciences

Åbo Akademi University
- Department of Computer Science
- Institute for Advanced Management Systems Research

ISBN 978-952-12-3054-7
ISSN 1239-1891