Investigating the Adoption of Metrics in Large-Scale Agile Software Development

Pascal Philipp
Technische Universität München
8 PUBLICATIONS **33** CITATIONS
SEE PROFILE

Investigating the Adoption of Metrics in Large-Scale Agile Software Development

Completed Research Paper

Pascal Philipp
Chair of Software Engineering for Business Information Systems (sebis), Department of Informatics, Technical University of Munich
Boltzmannstraße 3, 85748 Garching, Germany
pascal.philipp@tum.de

Franziska Tobisch
Chair of Software Engineering for Business Information Systems (sebis), Department of Informatics, Technical University of Munich
Boltzmannstraße 3, 85748 Garching, Germany
franziska.tobisch@tum.de

Florian Matthes
Chair of Software Engineering for Business Information Systems (sebis), Department of Informatics, Technical University of Munich
Boltzmannstraße 3, 85748 Garching, Germany
matthes@tum.de

Abstract

Many organizations have started using agile methods to develop software in small projects. The success of agile methods on a small scale inspired organizations to scale them to larger contexts. However, adopting agile practices at scale can complicate the monitoring, coordination, and steering of the multi-layered development process. Metrics can address this challenge but are controversial since their implementation is a challenge in itself. Hitherto, research on large-scale agile development lacks publications investigating the adoption of metrics. We conducted an expert interview study to explore their (i) reasons for adopting metrics (ii), what metrics they use on the team, program, and portfolio level (iii), and the most occurring metrics in the expert organizations. Our results show that metrics are mainly used for transparency, improvement, and controlling. Most metrics occur on the program level. Finally, we identified story point estimation, velocity, and sprint burn-down, as the most established metrics.

Keywords: Agile software development, large-scale agile, metrics, reasons

Introduction

Organizations act in an environment in which rapid changes in technology, markets, and regulations increasingly force them to adapt quickly to remain competitive (Preiss et al. 1996; Van Oosterhout et al. 2006). Simultaneously, products and services increasingly depend on software (Attaran 2004; Highsmith and Highsmith 2002; Van Oosterhout et al. 2006). In dealing with the growing demand for agility, traditional software development is ever more reaching its limits (Highsmith and Cockburn 2001; Highsmith and Highsmith 2002). Its severe weaknesses, such as infrequent releases with large batch sizes, a lack of customer feedback, and rigid processes, acutely restrict organizations (Javdani et al. 2012; Petersen and Wohlin 2009). These shortcomings of traditional software development gave rise to the success of agile software development (Javdani et al. 2012; Petersen and Wohlin 2009). Developing software in an agile manner has lastingly changed and coined existing software development practices (Dingsøyr et al. 2012; Rajlich 2006). Accordingly, agile approaches like Scrum (Schwaber and Beedle 2002) and Extreme Programming (XP) (Beck 2000) have been widely adopted by practitioners (Digital.ai 2020; Hamed and Abushama 2013; Maiden and Jones 2010). These approaches work well on a small scale in contexts close to the 'agile sweet spot' comprising ideal conditions like co-located teams of 5–12 members; a stable underlying architecture; medium to low system criticality, and frequent deliveries (Kruchten 2013; Nord et al. 2014). Inspired by the success of agile methods on a small scale, an increasing number of large organizations have started to scale agile practices up to larger contexts which depart significantly from the 'agile sweet spot' (Digital.ai 2020; Dikert et al. 2016; Dingsøyr et al. 2018; Kruchten 2013; Nord et al. 2014). Scaling agile methods is coined as large-scale agile software development (Dingsøyr and Moe 2014; Rolland et al. 2016). Large-scale agile software development can refer to settings where multiple teams work together on a software product (Dingsøyr and Moe 2014; Dumitriu et al. 2019). Further, large-scale agile software development comprises implementing agile methods and principles across a whole organization, including higher organizational levels (Dingsøyr and Moe 2014; Dumitriu et al. 2019). Several large-scale agile practices and frameworks, such as the Scaled Agile Framework (SAFe) (Leffingwell 2018), and Large Enterprise Scrum (LeSS) (Larman and Vodde 2016), have been developed to support organizations (Paasivaara 2017) applying agile approaches at scale (Dikert et al. 2016; Nerur et al. 2005).

The term '*metric*' is generally understood to mean measurement (Geras et al. 2004; ISO 2017). Measurement can be described as "the empirical, objective assignment of numbers, according to a rule derived from a model or theory to attributes of objects or events with the intent of describing them" (Geras et al. 2004). Metrics can serve managers as feedback instruments by providing concise quantitative information. Therefore, metrics can be seen as an alternative to or be combined with qualitative information (i.e., reports, group discussions, or interviews). In traditional software development, metrics have played an essential role for decades (Fenton and Neil 2000) and are used to measure different attributes (e.g., costs) of products, processes, and resources (Fenton and Bieman 2014). Several researchers argue that metrics implemented in traditional software development approaches are not always applicable for using them within agile settings (Hartmann and Dymond 2006; Oza and Korkala 2012; Poligadu and Moloo 2014). As measurements are essential drivers for behavior, there is the risk that using improper metrics negatively impacts team behavior, productivity and undermines the cultural change towards an agile way of working (Hartmann and Dymond 2006). However, within single agile teams, metrics like the burn-down chart (Schwaber 2004; Scrum.org 2021) or velocity (Scrum.org 2021) are applied and can support a key principle of agile software development: data-driven continuous feedback and learning (Janus et al. 2012; Liechti et al. 2017). In scaling agile environments, the increased organizational scope (Brown et al. 2013; Dingsøyr et al. 2018) and complexity (Ambler and Lines 2014; Brown et al. 2013) can lead to various challenges, such as the difficulty of monitoring, coordination, and steering of a multi-layered development process (Talby and Dubinsky 2009). Metrics can address this challenge because they allow to monitor, control, assess, manage, and improve the software process (Ebert et al. 2005; Jethani 2013). Moreover, metrics are means to identify problems and risks early, support communication, and enable fact-based planning, estimating decisions, and optimization (Jethani 2013). However, introducing metrics in scaling agile environments can be challenging itself. Numerous organizations struggle with efficiently adopting metrics (Korpivaara et al. 2021), due to difficulties in reducing fragmented project specific data and measurement (Digital.ai 2020), connecting metrics belonging to different organizational levels while considering level-specific, varied requirements and appropriate metrics scaling (Korpivaara et al. 2021). Besides these challenges specific to scaling agile environments, implementing metrics in general can lead to challenges like lacking credibility of measurements caused by inadequate data quality (Eckerson 2010), misunderstandings, and misinterpretations (Bird et al. 2005; Eckerson 2010). Metrics are well-researched in general management (cf., Kütz 2011; Mills 1988; Neely et al. 2002) and traditional software engineering (cf., Fenton and Neil 2000). On the contrary, expert insights investigating the adoption of metrics in scaling agile environments are scarce (Uludağ et al. 2021). Moreover, examining the usage of metrics beyond the team level was identified as a burning research topic (Dingsøyr and Moe 2014). Therefore, we intend to conduct an expert interview study to collect and analyze their reasons for using metrics in scaling agile environments. Moreover, we aim to observe on which scaling levels metrics occur and how important traditional metrics (i.e., stemming from general management) and agile metrics (i.e., related to agile concepts, events, and artifacts) are on these levels, according to the experts. Our final goal is to present the most mentioned metrics occurring in at least three expert organizations. Based on these goals, we formulate the three research questions (RQs):

- *RQ1: Why do practitioners adopt metrics in large-scale agile software development?*
- *RQ2: Which metrics do practitioners use on the team, program, and portfolio level?*
- *RQ3: Which metrics do practitioners use the most in large-scale agile software development?*

Theoretical Background

Large-scale Agile Software Development

Nowadays, applying agile methodologies in software development is the predominant mode of developing software (Dingsøyr and Moe 2014). Agile methods rely on the values and principles summarized in the agile manifesto (Beedle et al. 2010; Schwaber 2018), published in 2001 to provide “an alternative to documentation driven, heavyweight software development processes” (Beck et al. 2001). Multiple agile methods and frameworks, including Feature-Driven Development, Dynamic System Development Method (DSDM), and Crystal Methods, have been developed to help organizations implement agility (Uludağ et al. 2021). Presently, Scrum, ScrumBan, and hybrid approaches, like a combination of Scrum and XP, are the most widely used agile approaches in the industry (Digital.ai 2020). The advantages of agile methods like responsiveness to volatile customer requirements, flexibility, and resilience in altering environments are undisputed (Digital.ai 2020; Kumar and Bhatia 2012). However, according to Kruchten (2013), the success of agile methods depends on the context in which practitioners deploy them. Implementing agile methods ‘out of the box’ is likely to be successful in the ‘agile sweet spot’, a context identical or similar to the context in which agile methods have been initially created (Kruchten 2013). According to Nord et al. (2014), a favorable context is characterized by co-located teams of 5-12 members; a stable underlying architecture; medium to low system criticality; and frequent deliveries. Kruchten (2013) identified eight key factors to assess whether a given context departs from the sweet spot. The key factors are size, stable architecture, business model, team distribution, rate of change, system age, criticality, and governance (Kruchten 2013). Applying agile approaches without modification in unfavorable contexts may lead to failure (Kruchten, 2013). The success of agile methods within the ‘agile sweet spot’ inspired large organizations to scale agile methods and practices to larger settings (Digital.ai 2020; Dikert et al. 2016; Dingsøyr et al. 2018). The approach of scaling agile methods is often referred to as large-scale agile software development (Dingsøyr and Moe 2014; Rolland et al. 2016). In the last years, multiple frameworks have been proposed to support employing agile approaches at scale (Dikert et al. 2016; Paasivaara 2017; Turetken et al. 2017). Most frameworks are based on Scrum and lean principles (Uludağ et al. 2017). The Annual State of Agile Report by Digital.ai (2020) classified Scaled Agile Framework (SAFe), Scrum-of-Scrums, Disciplined Agile Delivery (DAD), Large Scale Scrum (LeSS), Enterprise Scrum (ES), and Lean Management as the most popular frameworks in the industry.

Metrics in Large-scale Agile Software Development

Previous research shows that managers require metrics to enhance their decision-making (Kütz 2011; Mills 1988; Neely et al. 2002). Adopting metrics can support organizations with steering, controlling, reporting, early warning, and identifying problems (Kaplan and Norton 1992; Kütz 2011; Neely et al. 2002). A well-known approach for adopting metrics within organizations is the balanced scorecard (BSC) by Kaplan and Norton (1992). In the domain of traditional software development, metrics have played an essential role for decades (Fenton and Neil 2000). Fenton and Neil (2000) use the term ‘software metrics’ to summarize a wide range of activities dealing with measurements in traditional software engineering. According to Fenton and Bieman (2014), software metrics are used to measure or predict attributes (i.e., measurable characteristics) of processes, products, or resources. Multiple researchers emphasize that metrics successfully applied in traditional software development are not always applicable within agile settings (Hartmann and Dymond 2006; Oza and Korkala 2012; Poligadu and Moloo 2014). A potential conflict between measurement in traditional and agile software development is that traditional approaches measure against a predefined plan, whereas agile approaches follow the value of embracing change (Korpivaara et al. 2021; Kupiainen et al. 2015). Kupiainen et al. (2015) identify the main differences of measurements between agile and traditional software development as empowerment of teams, customer focus, simplicity, orientation on trends, fast feedback, and responsiveness to change. The Annual State of Agile Report conducted by Digital.ai (2020) identified the most used metrics to measure the success of agile transformations within individual agile projects in the industry. Sorted according to their popularity, these metrics are business value delivered, customer satisfaction, velocity, budget vs. actual cost, and planned vs. actual stories per iteration (Digital.ai 2020). Likewise, the research investigated metrics in non-scaled agile software development environments. According to Kupiainen et al. (2015), the main reasons for using metrics in agile software development are planning and progress tracking for sprints and projects, understanding and improving quality, fixing software process problems, and motivating people. Hossain et al. (2021) provide an overview of agile software development process metrics and identify performance on delivery, cycle completion rate, and development speed as the most important high-level metrics. Aldahmash and Gravell (2018) propose an instrument for measuring the success of agile projects with the help of metrics. Moreover, some research was conducted to identify metric categories. Oza and Korkala (2012) assign metrics in agile software development to the categories Project Management, Engineer, Test, Automation, Strategy, Iteration, and Code. Another proposed approach is mapping metrics in agile software development to the Scrum Events Sprint Planning, Daily Scrum, Sprint Review, and Sprint Retrospective (Kurnia et al. 2018). Kupiainen et al. (2015) suggest a categorization of metrics in agile software development based on reasons and agile principles. Also, research has been conducted to investigate metrics in scaled agile software development environments. Several studies investigated which metrics are adopted by practitioners in general (Dubinsky et al. 2005; Kettunen et al. 2019; Ram et al. 2018; Storti and Clear 2020; Talby et al. 2006) and within measurement programs (Staron and Medig 2011). Further metrics are applied as an indicator for the success in large-scale agile software development based on goals (Shirokova et al. 2020) or to measure the success of the transformation (Kišš and Rossi 2018; Mikalsen et al. 2020; Olszewska et al. 2016). Moreover, metrics are used to measure team performance (Cheng et al. 2009; Korpivaara et al. 2021), to support (Talby and Dubinsky 2009), or report on the portfolio level (Stettina and Schoemaker 2018).

Methodology

Study Design and Overview

We conducted semi-structured expert interviews (Fontana and Frey 2000; Myers and Newman 2007; Seaman 1999) to answer our research questions. To ensure a rigorous collection of empirical evidence, we designed our research methodology based on the guidelines of Myers and Newman (2007) for semi-structured interviews. The chosen research design relies on qualitative data collection and is appropriate since adopting metrics in large-scale agile development is a problem in practice (Seaman 1999). The design of the interview questions was derived during a workshop from our research questions. This study has an exploratory character but also includes descriptive and explanatory elements. Table 1 presents an overview of the study participants. In total, we conducted 23 interviews with practitioners from thirteen organizations. The selection of the interviewees was intentional, and the purpose was to identify interviewees that work in environments typical for large-scale agile development. Large-scale agile software development refers to environments where multiple agile teams work together on a software product (Dingsøyr and Moe 2014; Dumitriu et al. 2019) or to organizations that apply agile methods across the entire organization, also on higher organizational levels. The expert organizations apply to this definition of large-scale agile software development, since within each organization multiple agile teams work together on a software product. Moreover, we only interviewed preferred roles, like Scrum Master, manager, or developer, to ensure enough topic-related knowledge. However, three interviews at one organization were excluded since the interviewees had no sufficient experience regarding the usage of metrics in large-scale agile software development. The included interviewees are characterized by a broad spectrum of job roles and industry backgrounds, leading to a large "variety of voices" covering multiple viewpoints (Myers and Newman 2007).

Data Collection

The interviews were conducted via videotelephony between January and March 2021. On average, the interviews had a length of 82 minutes, resulting in a large amount of empirical data totalling 1.640 minutes of audio recordings. In particular, the interview length depended on the extent of metric usage within the organizations. The interview durations exclude the time at the beginning of each expert interview required for the introduction and clarifications. Whereas the introduction covered the research goal, the clarifications addressed the interview structure and the permission to record the interview. At least two researchers guided each interview to enhance observer triangulation (Runeson and Höst 2009). All interviews followed the same outline, and the questions did not change throughout the course of the interviews. However, due to the interviews' semi-structured nature, the questions were asked in changing the order, or slight variations of the questions' wording did occur. First, questions regarding the professional background and the large-scale agile environment at the corresponding organization were asked. It is worth mentioning that some of the experts answered the questions related to the first part of the questionnaire before the actual interview. Subsequent questions targeted the adoption of metrics (e.g., reasons for using metrics and applied metrics at the organization). In addition to the interviews, we included relevant files shared by our interview partners to facilitate the triangulation of data sources. The results presented in this study are a subset of the overall collected data. We plan to communicate the remaining evidence in separate publications.

No.	Role(s), Large-scale agile software development experience (years)	No.	Role(s), Large-scale agile software development experience (years)
E1	Agile Coach, 11-15	**E11**	Manager, 3-5
E2	Manager, 3-5	**E12**	Scrum Master and Agile Coach, 3-5
E3	Manager, 6-10	**E13**	Agile Coach, 6-10
E4	Scrum Master and Agile Coach, 1-2	**E14**	Scrum Master and Agile Coach, 3-5
E5	Manager and Agile Coach, 6-10	**E15**	Release Train Engineer, 1-2
E6	Developer and Scrum Master, 3-5	**E16**	Scrum Master and Agile Coach, 1-2
E7	Product Owner and Business Analyst, 1-2	**E17**	Manager and Agile Coach, 3-5
E8	Software Architect, Solution Architect, 3-5	**E18**	Agile Coach, 3-5
E9	Scrum Master, Agile Coach and Developer, 11-15	**E19**	Scrum Master and Agile Coach, 6-10
E10	Software Architect, 11-15	**E20**	Release Train Engineer, 6-10

Table 1. Interview Partners

Data Analysis

Two researchers transcribed the audio recordings, and another researcher ensured quality by reviewing the completeness and consistency of each transcript. We guaranteed the anonymity of the interviewees by dismissing sensitive information. In addition, we assigned unique pseudonyms to refer to the experts and organizations throughout this paper. We combined deductive and inductive coding (Miles et al. 2020) to code the interview transcripts, the pre-filled questionnaire parts, and the additional files. The coding was performed following the two-cyclic approach described by Saldaña (2021). First, we assigned descriptive (Miles et al. 2020) first-cycle codes to text segments, which in turn were assigned to the higher-order second-cycle codes. For the second-cycle deductive coding, we used an initial list of higher-order codes (Saldaña 2021) as a starting point. Our list was aligned with the questions of the interview questionnaire. We created additional second-cycle codes inductively whenever new findings were made while analyzing or a text fragment could not be assigned to an existing code category (Saldaña 2021). The introduction of new codes required the re-codification of the previously coded data. Consequently, the final coding was developed incrementally. We used MAXQDA[1] and Microsoft Excel[2] to code and analyze the transcripts. We discussed and resolved conflicts by mutual consent to increase the result validity. Whenever the interpretation of the evidence required clarification, we contacted the interviewees again to resolve ambiguities.

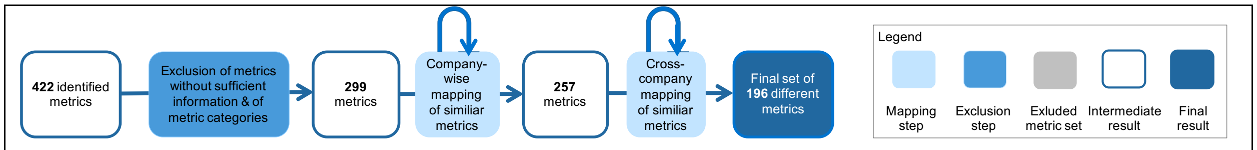
422 identified metrics
Exclusion of metrics without sufficient information & of metric categories
299 metrics
Company-wise mapping of similiar metrics
257 metrics
Cross-company mapping of similiar metrics
Final set of 196 different metrics
Legend
Mapping step
Exclusion step
Exluded metric set
Intermediate result
Final result

Figure 1. Metric Filtering and Mapping Steps

In the empirical data, we identified an initial set of 422 potential metrics. Based on predefined exclusion criteria, we reduced the initial set of metrics to increase the quality and relevance of the finally included metrics. An overview of the filtering and mapping process is provided in Figure 1. Whenever experts just mentioned a category of metrics like "financial KPIs" (E2) or "code quality metrics" (E14) instead of a specific metric, we did not count the category as a metric. In total, we excluded 78 metrics that did not meet our quality requirements due to insufficient information regarding their organizational level of usage, calculation, and cadence necessary for a profound understanding. From the remaining 299 metrics, we excluded 103 duplicates within the same organization and between the organizations resulting in 196 finally included metrics.

Level	Description
Team level	The organizational scope is a single scrum team with around nine team members and/or a team with a single team backlog shared by all team members.
Program level	The organizational scope is multiple teams that develop one product and/or develop in the same cadence/cycle and/or are grouped into an organizational unit like an ART or a solution train. Moreover, this level covers multiple groups of teams that develop sub-products of one large product and topics directly related to a single product or large solution.
Portfolio level	The organizational scope is a group of programs.
Organizational level	The organizational scope is multiple portfolios, topics related to the corporate strategy influenced by government and regulations, as well as organizational topics that relate to the overall organization and are not directly related to the program and portfolio structures.

Table 2. Categorization of Organizational Levels in the Context of Large-Scale Agile Software Development Inspired by Korpivaara et al. (2021) and Stettina and Schoemaker (2018)

We classified each of the 196 included metrics as either agile or traditional metrics. At least one of the following criteria had to be fulfilled that a metric was classified as an agile metric:

- Calculation based on agile concepts such as story points or user stories (Cohn 2005)
- Measurement of aspects related to continuous integration and delivery (Stolberg 2009)
- Relation to agile events such as the sprint planning (Schwaber and Sutherland 2020), also on a scaled level
- Relation to agile artifacts like the product or sprint backlog (Schwaber and Sutherland 2020), also on a scaled level

The remaining metrics, which did not fulfill any of the mentioned criteria for agile metrics, were classified as traditional metrics.

[1] https://www.maxqda.com

[2] https://www.microsoft.com/sv-se/microsoft-365/excel

Based on the categorization developed by Korpivaara et al. (2021), Table 2 describes the levels we used to assign the identified metrics to the team, program, portfolio, and organizational level. It is worth mentioning, that the expert environments were heterogeneous and different terms were used to describe the same level. Consequently, we performed a mapping between the levels mentioned in the expert environment and our categorization schema.

Results

Overview of the Experts' Development Organizations

Table 3 provides an overview of the of the experts' development organizations.

Company	Description of the experts' development organization
CarCo1	The company's development group is at the beginning of its SAFe transformation and aims to implement the Full SAFe configuration. The programs develop software products for the company's end-users and solutions for internal stakeholders. Within the program of E1, 750 employees work together.
CarCo2	The development organization of E12 uses Scrum@Scale and plans a transition to LeSS in the mid-term future. E12 was not allowed to provide information regarding the implemented software solution. Within the program, approximately 100 employees work together.
ConsultCo	The program of E4, E6, E7, E8 and E11 implements a customized scaling agile approach similar to LeSS. The approach also includes Scrum-of-Scrum elements. Within the program, 50 employees in different teams work together to develop a software solution for a client. The client implements a portfolio level to keep track of all projects conducted with ConsultCo.
BankCo	The development organization of E2 and E18 uses a custom framework similar to SAFe and consists of around 6.000 people grouped into 18 higher-level key areas. Whereas E2 is responsible for the enterprise architecture, E18 works in one of the key areas of the development organization consisting of 400 employees.
HealthCo	The company has multiple heterogeneous development groups using different frameworks. The program of E3 and E5 implements a custom lean approach combining elements of LeSS and SAFe. The program develops software for a medical product that is sold to end customers. Within the program, approximately 450 employees are working together. The development organization of E10 implements Scrum-of-Scrums and develops software for internal stakeholders. Within this program, approximately 500 employees are working together.
InsureCo1	The company's whole IT department is organized in an agile but heterogeneous way. E14 is working in an area implementing SAFe. The program of E14 consists of 75 employees that develop software for the business line health insurances in the area of sales and operation. E17 is working in a development area implementing Nexus to develop software for non-personal non-life insurance. E17's specific program consists of around 90 people. E15 is working in a development area that is using SAFe as well. Within the program of E15, 80 employees develop a cross-sectional solution and several smaller related sub-products used by company-internal customers.
InsureCo2	The development organization of E16 is implementing Scrum@Scale and comprises approximately 35 employees. The program intends to develop software that supports insurance processes.
RetailCo1	The development group of E13 uses a custom framework combining elements of SAFe, Spotify, and Scrum-of-Scrums and comprises 600 people. E13's program consists of 120 people intending to develop a warehouse management system.
RetailCo2	The company's international IT department applies different scaling agile frameworks. Within the development area of E19 work around 3.700-4.700 employees. The program of E19 uses elements of LeSS, Scrum-of-Scrums, and Spotify. Within this program, in total, about 80 internal employees develop a software solution platform in the area of sales.
TeleCo	The development organization of E9 uses elements of LeSS and Scrum-of-Scrums. The program comprises 100 developers grouped into eight teams that develop a software solution to support the implementation of the communication standard Global System for Mobile Communications (GSM).
Transport Co	The development group of E20 implements a Full SAFe configuration and incorporates elements of other frameworks. In total, within the program of E20 approximately 500 people are working from two different subsidiaries of the parent organization. Also, external employees are involved in the program. They work together on software products in the context of sales and ticket booking.

Table 3. Overview of Expert Organizations

Reasons for Using Metrics in Large-Scale Agile Software Development

To answer RQ1, we asked for the experts' reasons for using metrics in their large-scale agile software development environment. In total, we identified and categorized 237 such reasons. Table 4 provides an overview of the reason categories and the contexts each reason category is referring to. The most frequently mentioned reason categories were *Transparency (39%), Improvement (22%), Controlling (13%), Planning (8%), Employee motivation (4%), Agility (3%), Problem identification (3%),* and *Stakeholder involvement (3%).* On a more detailed level, each reason category can refer to one of the following contexts: *Development organization and processes, Product, Entire organization, Customer, Finance, Employee, and Stakeholder*. For example, the experts in our study mentioned the reason of increasing Transparency regarding the context *Development organization and processes* 56 times. Whenever a reason category was mentioned, but the expert did not clearly reference it to a context, we assigned the reason to the context *Generic*. Subsequently, we describe each reason category in more detail.

Transparency

In the *Development organization and processes*, metrics are applied to enhance transparency regarding the efficiency of the processes and development environments (E3, E19) on the program (E4, E8, E10, E11, E17) and team levels (E6, E17). For instance, E15 mentioned that the number of features per program increment appears to be constrained by achieving transparency on potential problems in programs. Release (E9) and epic burn-downs (E12) are utilized to clarify the program's progress (E9, E12) and deviations from its plan (E12). Likewise, at the team level, the sprint burn-down is used to create transparency as a basis for self-improvement (E16). According to expert E3 the sprint burn-down helps to *"to stimulate the discussion and create transparency"*. Velocity is predominately implemented on the team level to gain insight into the teams' progress (E17), stability (E1), its performance (E17), and to identify potential problems (E17). Regarding the *Product* context, metrics are used to create transparency with the respect to the value created by developing products (E19) and their performance when used by customers (E3). Expert E3 stated that creating transparency early is beneficial since *"[...] the earlier we find those topics, the easier we can do something against it"*. For example, on the portfolio level, the delivery rate of quarterly business review epics was introduced to create transparency on the actual delivery (E18). According to E9, the release burn-down is adopted to identify the need for de-scoping of the product's functionality. Measuring the total defects is done to gain transparency on the stability of a product (E8). Expert E8 made clear that they do not count the defects to blame the teams: *"[...] we don't try to blame the teams [...] it's more like we try to generally aggregate how good we are] "*. Additionally, the defect-feature ratio is implemented to assess product quality (E1). Finally, the metric feature usage is measured to make bad investments into features transparent (E1). Some experts mentioned transparency as a reason for using metrics (E5, E6, E7, E8, E12, E13, E15) without specifying a particular context. Metrics are used to increase transparency regarding the current state (E4), progress (E17), problems, and the need for actions (E2) and reactions (E2). The experts also use metrics to enhance transparency within the *Entire organization* (E18). According to E18 this especially important within large organizations: "[...] *we have 7,000 people working in this agile environment [...] to get really an overview where [...] we actually stand on the highest level, on the delivery organization level you need [...] metrics."* Metrics are used to make the organizations' progress (E2, E18) and current situation (E1, E8, E12, E13) transparent. Within the *Customer* context, metrics are used to enhance transparency on customer-related topics. For example, the release burn-down on the program level is used for realistic customer offers based on detailed planning information (E9). Further, improving the transparency regarding customer satisfaction is one main reason for adopting customer satisfaction metrics (E6, E8). Inside the *Finance* context, metrics are measured to contribute towards financial transparency (E6), for instance, on revenue (E13). Additionally, metrics are used as early indicators for economic success (E11) and to create transparency regarding the profitability of a program (E7). In the *Employee* context achieving transparency of employee-related topics is a reason to implement metrics. For example, employee satisfaction metrics (E6) are implemented to upturn transparency of the employee (E6, E8) and team satisfaction (E6). Within the *Stakeholder* context, metrics are used to surge transparency regarding stakeholders. An example metric is measuring the transparency of the performance of collaboration partners (E19).

Improvement

Metrics are used to improve the efficiency (E3, E19, E20) and team performance (E6) in the program within the *Development organization and processes*. Further metrics serve as the basis for the self-improvement of teams (E2, E6). Sprint report metrics like the number of remaining story points after a sprint help teams improve regarding their way of working and their commitment (E16). The metric planned team velocity is used to avoid overloading the team with work items and thus improve the team's management (E17). To improve the stability of the development flow of programs, the metric feature lead time is applied (E1). Expert 1 stated that using the metric feature lead time is helpful because the *"biggest advantage is we can stabilize [...] the flow of production and that is crucial for any success. If you have a chaotic system you won't be successful. [...] And so you have to care about the flow of production development. And so, that's the biggest advantage, when we are able to measure that right"*. The experts also use metrics to achieve improvements in the *Product* context. Particularly metrics are used to improve the quality of the product (E17). The purpose of measuring test errors is to improve the developer's analytical capabilities and achieve more stable software (E5). Metrics for measuring the current product performance or stability are used to improve product competitiveness (E3). Additionally, customer satisfaction is measured to improve the quality of features (E14). Whenever experts mentioned improvement as a reason for using metrics (E3, E14, E18) without specifying a specific context, we assigned the reason to the context *Generic*. Metrics are implemented since they support the identification of improvement potentials (E4, E8, E11, E18). Further, metrics are used to better reach expectations (E9) and goals (E3). Also, metrics are a means to enhance comparability, which supports the identification of best practices (E3). Within the context *Entire organization,* metrics are implemented to contribute to organization-wide improvement. For instance, metrics are used to initiate organizational change (E3) or stabilize the overall organization (E1). Managers also use metrics to enrich the reporting towards defined goals (E1). In the context *Employee,* metrics are adopted to improve employee-related topics like employee satisfaction. Therefore, metrics can be the basis for detecting improvement potentials regarding employee and team satisfaction (E6). Also, metrics can be used to improve the managers' awareness of teams' health (E17) and promote employees' self-responsibility (E15). In the context *Customer,* metrics are mainly used to enhance customer satisfaction. An example is the metric defects in production, which, when minimized, helps improve customer satisfaction (E9).

Controlling

Within the *Development organization and processes*, metrics are used to control the progress of the programs (E20). For example, the metrics release burn-down or number of features per program increment are measured for steering programs in the right direction (E15). Within the program, the metric release burn-down is used to control the complex development processes (E3, E12). The metric velocity is used to steer teams (E16), and the number of completed user stories of a program epic enables the controlling of development goals. Whenever experts mentioned controlling as a reason for using metrics *(E3, E4, E7, E10, E12, E14)* without specifying a particular context, we assigned the reason to the context *Generic*. Expert E3 explained why using metrics to support controlling is important: *"So, for me one of the observations, which is already ten years old, you do not control what you cannot measure"*. In the context *Product,* metrics are used for product-related controlling. For example, the metric total defects enable the controlling of defects (E4). Also, within the *Entire organization,* metrics are used for controlling. For example, metrics are used to control the overall organization (E1) and steer the organization in an agile way (E1). Managers also use metrics to control the achievement of personal goals or bonuses (E1). In the context *Employee,* metrics are applied to control the employees working in an organization (E1, E2). As stated by expert E2 metrics are a suitable instrument to control employees because "*Well, the thing is that the metrics [...] make people do the right things"*. Furthermore, metrics are implemented to control employee and team satisfaction as well as team health (E6). As stated by expert E6 controlling is *"also about the team satisfaction, to control [...] the team health, how the situation looks like, how to avoid some problems to coordinate some atmosphere in the team, so this also is really important."* In the *Customer* context, metrics are used, for example, to control the competitiveness based on price (E19).

Planning

Metrics such as the remaining story points are used for enabling planning and avoiding over-planning (E16) in the *Development organization and processes*. The metric velocity on the team level is measured to assist sprint capacity planning and to enhance the planning of the program's roadmap (E16). Experts use the program velocity factor per sprint to support the planning in programs (E7, E11), for example, regarding milestones (E8). With the help of the lead time of large initiatives on the program level, the initiatives are prioritized (E13). Whenever experts mentioned planning as a reason for using metrics (E7, E8, E11, E13) without specifying a particular context, we assigned the reason to the context *Generic*. Further, metrics are adopted since they can improve predictability in general (E5). In the *Product* context, metrics are used to support the planning of product-related topics such as planning of delivery roadmaps (E16). Further, metrics are implemented to improve the delivery predictability (E6, E10, E16, E17, E20). Accordingly, expert E17 stated: *"forecasting delivery dates is still a thing. We need to do it, there is no way around it"*. Moreover, expert E20 highlighted the need to improve the delivery predictability: *"[...] we have a hard deadline. We want to replace our existing system at the end of 2022. What we want to do now with the* *metrics is to see if our velocity is fast enough".* The metric remaining defects per sprint in a program is measured to support the capacity planning to fix a product's defects (E11).

Reason (Occurrences)	Experts	# Experts
Transparency (111, 39%)		
Development organization and processes, (56, 20%)	E1, E2, E3, E4, E6, E7, E8, E9, E10, E11, E12, E13, E15, E16, E17, E18, E19, E20	18 (90%)
Product, (18, 6%)	E1, E2, E3, E8, E9, E11, E14, E15, E18, E19	10 (50%)
Generic, (13, 5%)	E2, E4, E5, E6, E7, E8, E9, E12, E13, E15	10 (50%)
Entire organization, (9, 3%)	E1, E2, E8, E10, E12, E13, E18, E20	8 (40%)
Customer, (7, 2%)	E1, E7, E8, E9, E14, E17	6 (30%)
Finance, (5, 2%)	E1, E6, E7, E9, E13	5 (25%)
Employee, (2, 1%)	E6, E8	2 (10%)
Stakeholder, (1, <1%)	E19	1 (5%)
Improvement 62 (22%)		
Development organization and processes, (28, 10%)	E1, E2, E3, E5, E6, E10, E12, E14, E16, E17, E19, E20	12 (60%)
Generic, (13, 5%)	E3, E4, E7, E8, E9, E11, E14, E15, E18	9 (45%)
Product, (12, 4%)	E1, E3, E5, E10, E11, E14, E15	7 (35%)
Entire Organization, (4, 1%)	E1, E3	2 (10%)
Employee, (4, 1%)	E6, E13, E15, E17	4 (20%)
Customer, (1, <1%)	E9	1 (5%)
Controlling (38, 13%)		
Generic, (14, 5%)	E3, E4, E7, E10, E12, E14, E15	7 (35%)
Development organization and processes, (10, 3%)	E2, E3, E5, E6, E12, E15, E16, E20	8 (40%)
Entire organization, (5, 2%)	E1, E12	2 (10%)
Employee, (4, 1%)	E1, E2, E6	3 (15%)
Product, (3, 1%)	E4, E7, E10	3 (15%)
Finance, (2, 1%)	E19	1 (5%)
Planning (24, 8%)		
Development organization and processes, (12, 4%)	E3, E4, E7, E8, E11, E13, E16, E17	8 (40%)
Generic, (7, 2%)	E5, E7, E8, E11, E13	5 (25%)
Product, (6, 2%)	E6, E10, E11, E16, E17, E20	6 (30%)
Employee motivation (11, 4%)		
Development organization and processes, (9, 3%)	E1, E4, E6, E7, E8, E15, E16, E17, E20	9 (45%)
Entire organization, (2, 1%)	E13, E15	2 (10%)
Agility (9, 3%)		
Generic, (8, 3%)	E2, E7, E12, E13	4 (20%)
Development organization and processes, (1, <1%)	E13	1 (5%)
Problem identification (9, 3%)		
Generic, (3, 1%)	E4, E12, E17	3 (15%)
Product, (3, 1%)	E7, E11, E12	3 (15%)
Development organization and processes, (3, 1%)	E6, E12, E20	3 (15%)
Stakeholder involvement (8, 3%)		
Customer, (6, 2%)	E6, E8, E11, E14, E15, E16	6 (30%)
Generic, (1, <1%)	E14	1 (5%)
Entire organization, (1, <1%)	E16	1 (5%)

Table 4. Reasons for Using Metrics in Large-Scale Agile Software Development

Employee Motivation

In the *Development organization and processes* metrics are used to enhance the motivation of the employees. Metrics like team velocity (E17) or sprint burn down (E1, E16) are measured to boost motivation. Using the sprint burn down chart as a motivator is described by expert E1: *"[...] the burn down chart is very nice if you have it on a task level, because it is a big motivator. When it's very prominent in the team space [...] and every time one task is finished it goes down it's a very big motivator".* The program velocity can strengthen the motivation of employees working in the program (E20). In the context *Entire organization,* metrics are used to encourage and inspire the workforce (E13) and ensure its commitment (E15).

Agility

Most experts mentioned *Agility* as a reason to use metrics without specifying a specific context (E2, E7, E12, E13). Metrics are implemented to enable (E2) and foster reactivity (E2, E7, E12). Expert E12 highlighted the importance to communicate that *"[...] it's mostly about trust and making the people understand you don't want to control them, you just want to understand on a higher level what's going on and be able to react when there are problems""*. Further, adopting metrics allows adaption to change (E7), including the adjustment of goals and strategy (E13).

In the Development organization and processes, metrics are used to increase agility by implementing measures such as velocity to react, replan, and perform readjustments of goals and the strategy in programs (E13).

Problem Identification

Most experts mentioned *Problem identification* as a reason for implementing metrics without specifying a specific context (E4, E12, E17). Thus, we assigned these occurrences to the context *Generic*. In the context *Product,* metrics are implemented to identify product-related problems. For instance, the metric completed user stories of an epic on program level can indicate delivery delays (E12). The metric total reported defects is measured to reveal problems regarding the product's usability (E11). In the *Development organization and processes,* metrics are used to indicate problems in programs. Therefore, for example, the metrics planned vs. actual program velocity (E20) and burn-down on program level (E12) are measured.

Stakeholder Involvement

Metrics are used to facilitate customer communication (E5, E8, E16) and joint planning (E5, E16). For example, to support customer communication and contract negotiation, the metric program velocity factor per sprint is implemented. The metrics number of defects in production is measured since it positively affects the relationship with the customers (E15). One expert (E14) mentioned supporting customer involvement as a reason without assigning it to a specific context. In the *Entire organization,* metrics are used to facilitate organization-wide stakeholder communication. For example, the epic burn-down on the program level is measured since it can support management communication (E16).

Further Reasons

Further reasons for using metrics are *Risk mitigation, Decision making, Goal setting, Creation of trust, and Coordination*. For example, metrics as the planned vs. actual program effort are implemented to mitigate product-related risks by facilitating early escalations (E15). Further, metrics are used to support *Decision making* since they provide relevant information (E5, E14). Metrics are also used for *Goal setting* because they can define goal targets in the form of lagging indicators (E9) and provide information regarding the status of goal-fulfillment (E2). Metrics, such as the release burn-down, can also facilitate *Trust* between the product management and teams of a program (E9). A final reason for using metrics is *Coordination*. In the *Development organization and processes,* metrics are used to support the coordination of multiple agile teams (E12).

Agile and Traditional Metrics on Organizational Levels

We were also curious about the organizational levels (i.e., team, program, portfolio, organizational) on which metrics are implemented and their type (i.e., traditional, or agile). Our results are not of quantitative nature as we do not involve any mathematical models. We rather intend to count the number of occurrences of metrics at the expert organizations within different categories (see Figure 2). We identified most metrics (140, 71.43%) on the program level, followed by the team level (42, 21.43%) and organizational level (9, 4.59%). The least metrics were found on the portfolio level (5, 2.55%). On all levels except the organizational level, agile metrics were rather used than traditional metrics. The dominance of agile metrics in comparison to traditional metrics is very prominent on the team level. Out of the 42 implemented metrics on the team level, are 38 metrics of agile nature (90.48%). On the organizational level, we only identified traditional metrics.

Most Used Metrics Among the Expert Organizations

To identify the most used metrics among the expert organizations, we asked the experts which metrics they use within their organization.

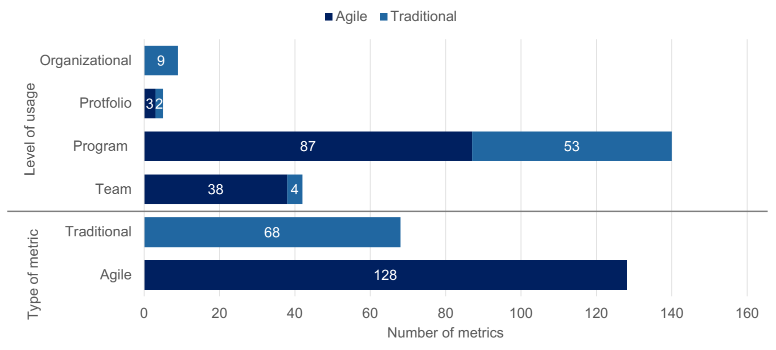

Figure 2. Distribution of Metrics per Organizational Level of Usage

Metric*			Organizations	#org.
Story point estimation: *Sum of the story points estimated for a user story.*	A	T	CarCo1, CarCo2, ConsultCo, HealthCo, InsureCo1, InsureCo2, RetailCo1, RetailCo2, TeleCo, TransportCo	10 (91%)
Velocity: *Sum of the story points of all user stories completed (Done) by a team in a sprint.*	A	T	BankCo, CarCo1, CarCo2, ConsultCo, HealthCo, InsureCo1, InsureCo2, RetailCo1, RetailCo2, TransportCo	10 (91%)
Sprint burn-down: *Sum of the story points of all Open user stories in a sprint of a team in relation to the time left in the sprint reduced for each completed (Done) user story.*	A	T	BankCo, CarCo1, CarCo2, HealthCo, InsureCo1, InsureCo2, TransportCo	7 (64%)
Defects in production: *Number of defects in the production environment(s) of the software product for each severity category.*	TD	P	ConsultCo, HealthCo, InsureCo1, TeleCo	4 (36%)
Planned velocity: *Sum of the story points of all user stories planned to be completed by a team in a sprint.*	A	T	ConsultCo, HealthCo, RetailCo2, TransportCo	4 (36%)
Program velocity per sprint: *Sum of the story points of all user stories completed (Done) by all teams in a program in a sprint.*	A	P	ConsultCo, HealthCo, RetailCo2, TransportCo	4 (36%)
Number of features per PI: *Number of features realized (Done) in a PI by a program.*	A	P	CarCo1, HealthCo, InsureCo1	3 (27%)
Planned program velocity per sprint: *Sum of the story points of all user stories planned to be completed by all teams in a program in a sprint.*	A	P	ConsultCo, RetailCo2, TransportCo	3 (27%)
Release-burn-down: *Sum of the story points of all user stories planned for a release that still need to be realized (Done) by all team in a program in relation to the time left until the release.*	A	P	HealthCo, InsureCo1, TeleCo	3 (27%)
Test coverage: *Percentage of the number of lines of code all tests currently execute in relation to the total number of lines of code of the software that is to be tested.*	TD	P	ConsultCo, InsureCo1, RetailCo2	3 (27%)

Table 3. *T = Team, P = Program; A = Agile, TD = Traditional

Similar to the section before we do not follow a quantitative research strategy to answer this research question. According to the *rule of* three (Coplien and Alexander 1996) we added a metric to Table 3 of the most used metrics whenever it was implemented in at least three organizations. In most organizations (10, 91%), the metrics *Story point estimation* and *Velocity* are measured. It is noteworthy that both metrics are implemented on the team level. The second most used metric is the *Burn-down chart* implemented in seven organizations (64%). The third most used metrics are *Defects in production, Planned velocity,* and *Program velocity per sprint,* which are implemented in four organizations (36%). It is noteworthy that most metrics (166, 84.69%) are custom metrics since they were found in only one organization.

Discussion

Four key findings emerge from this expert interview study. First, metrics are mainly used to increase the transparency within the development organization on the program and team level. This observation is in line with Ebert et al. (2005) and Jethani (2013), who argue that metrics can be used to monitor, control, assess, manage, and improve the software process. Similar to Kupiainen et al. (2015) finding that metrics are used for progress tracking, we observed that the experts adopted release and epic burn-downs to clarify the program's progress and deviations from its plan. In contrast to Basili et al. (1994; 2010), who is highlighting the importance of aligning metrics with goals, using metrics for goal-setting was among the least mentioned reasons. Only a few experts revealed to use metrics to define goal targets and provide information regarding the status of goal-fulfillment. Moreover, our finding suggests that in scaling agile environments, metrics are rather adopted to fulfill traditional use cases from general management like steering, controlling, reporting, early warning, and identifying problems (Kaplan and Norton 1992; Kütz 2011; Neely et al. 2002). Second, on the team level, mainly agile metrics are used. We derive from this finding that the organizations tend to refrain from implementing traditional metrics on the team level, since they can undermine the cultural change towards an agile way of working (Hartmann and Dymond 2006). This finding is also backed by our third key finding revealing that the organization's most-used metrics are the story point estimation of user stories, velocity, and sprint burn-down. This finding corresponds with the results of Kupiainen et al. (2015), who identified velocity and effort estimates as the most important agile industrial metrics. Moreover, this finding is in line with the finding of Schwaber (2004), that within single agile teams, metrics like the burn-down chart or velocity are applied. Likewise, the Annual State of Agile Report by Digital.ai (2020) presents velocity and iteration burn-downs as the most used metrics in the industry. Fourth, most organizations implement metrics on the program level. This finding is in line with Kettunen et al. (2019), who reported that program metrics such as the cycle time, release cycle, lead time of features and epics, and defects are the most frequently used metrics. Further, this finding is in line with previous research showing that managers, which act on higher organizational levels, require metrics to enhance their decision-making (Kütz 2011; Mills 1988; Neely et al. 2002).

This study's results contribute to research on large-scale agile software development. To the best of our knowledge, no research exists investigating reasons for metrics in scaling agile environments. The identified reasons and their associated contexts can serve practitioners to identify usage scenarios for metrics in their organization. Further, we contribute to existing research by highlighting that agile metrics are frequently used on the program level and by identifying the most used metrics among the expert organizations. Consequently, practitioners working in scaling agile environments with low metric maturity regarding the number of implemented metrics should consider adopting agile metrics beyond the team level on the program level. Further, they can select metric candidates from the most used metrics that seem to have a convincing cost-benefit ratio since most expert organizations use them.

We apply the assessment schema of Runeson and Höst (2009) in this study to evaluate possible threats of validity. Since our expert study has an explanatory character and does not aim to determine casual relationships, internal validity is not a considered as a threat. The concern of construct validity is whether the research design is appropriate to investigate our three research questions. We addressed this threat by including several sources during the data collection. In addition to semi-structured interviews with experts working in different roles and industries, we incorporated internal documents of the development organizations in our research. External validity relates to the generalization of our findings. Since we clearly outline how our results relate to the experts and organizations of our study, we effectively countermeasure this threat. Reliability is achieved through the rigorous conduct of this study and by ensuring that a replication of our study would yield the same results (i.e., researchers bias). To counteract researchers bias, three countermeasures were taken. First, at least two researchers were present during each interview. Second, all transcripts were reviewed by a second researcher. Third, whenever the interpretation of the evidence required clarification, we contacted the interviewees again to resolve ambiguities.

Conclusion

Industry and research show an increased interest in scaling agile methods beyond the team level (Digital.ai 2020; Dikert et al. 2016). However, scaling agile methods leads to increased organizational scope as well as complexities (Brown et al. 2013). Adopting metrics can help in managing those challenges (Brown et al.

2013; Talby and Dubinsky 2009). Currently, in the extant literature, publications regarding the application of metrics in scaling agile environments, investigating the reasons for using metrics, and studying which kind of metrics practitioners implement and find important is lacking. Against this backdrop, we conducted an expert interview study comprising 20 experts from 11 organizations and identified 196 metrics and 237 reasons. The identified metrics are not domain specific and can be applied independent of the industry context of an organization. We classified the identified metrics as agile or traditional, identified their organizational level of occurrence, and applied the 'rule of three' to derive reoccurring metrics. Further, we determined eight main reasons for using metrics in large-scale agile software development endeavors and described in detail in which contexts those reasons occur. Finally, this paper leaves some room for future research. First, we aim to publish the remaining evidence regarding goals, challenges, success factors, advantages, and disadvantages of using metrics in large-scale agile development endeavors. Second, we would like to publish all the identified metrics in a metric catalog based on a structured metric fact sheet. Also, we would like to investigate the implementation of the metric catalog within organizations.

References

Aldahmash, A. M., and Gravell, A. 2018. "Measuring Success in Agile Software Development Projects: A Gqm Approach," *Proceedings of the 13th International Conference on Software Engineering Advances (ICSEA)*, Nice, France: IARIA, pp. 38-44.

Ambler, S., and Lines, M. 2014. "Scaling Agile Software Development: Disciplined Agility at Scale," in: *Disciplined Agile Consortium White Paper Series*.

Attaran, M. 2004. "Exploring the Relationship between Information Technology and Business Process Reengineering," *Information & management* (41:5), pp. 585-596.

Basili, V. R., Caldiera, G., and Rombach, H. D. 1994. "The Goal Question Metric Approach," in: *Encyclopedia of software engineering*. pp. 528-532.

Basili, V. R., Lindvall, M., Regardie, M., Seaman, C., Heidrich, J., Münch, J., Rombach, D., and Trendowicz, A. 2010. "Linking Software Development and Business Strategy through Measurement," *Computer* (43:4), pp. 57-65.

Beck, K. 2000. *Extreme Programming Explained: Embrace Change*. Addison-Wesley Professional.

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., and Jeffries, R. 2001. "Manifesto for Agile Software Development."

Beedle, M., Coplien, J. O., Sutherland, J., Østergaard, J. C., Aguiar, A., and Schwaber, K. 2010. "Essential Scrum Patterns." Retrieved 11. November, 2019, from https://www.hillside.net/plop/2010/papers/beedle.pdf

Bird, S. M., Sir David, C., Farewell, V. T., Harvey, G., Tim, H., and Peter C, S. 2005. "Performance Indicators: Good, Bad, and Ugly," *Journal of the Royal Statistical Society: Series A (Statistics in Society)* (168:1), pp. 1-27.

Brown, A. W., Ambler, S., and Royce, W. 2013. "Agility at Scale: Economic Governance, Measured Improvement, and Disciplined Delivery," *Proceedings of the 35th International Conference on Software Engineering (ICSE)*, San Francisco, CA, USA: IEEE, pp. 873-881.

Cheng, T.-H., Jansen, S., and Remmers, M. 2009. "Controlling and Monitoring Agile Software Development in Three Dutch Product Software Companies," *ICSE Workshop on Software Development Governance*, Vancouver, BC, Canada: IEEE, pp. 29-35.

Cohn, M. 2005. *Agile Estimating and Planning*. Pearson Education.

Coplien, J. O., and Alexander, A. W. O. 1996. "Software Patterns."

Digital.ai. 2020. "14th Annual of State of Agile Report." Retrieved 29. January, 2022, from https://www.qagile.pl/wp-content/uploads/2020/06/14th-annual-state-of-agile-report.pdf

Dikert, K., Paasivaara, M., and Lassenius, C. 2016. "Challenges and Success Factors for Large-Scale Agile Transformations: A Systematic Literature Review," *Journal of Systems and Software* (119), pp. 87-108.

Dingsøyr, T., and Moe, N. B. 2014. "Towards Principles of Large-Scale Agile Development," *Proceedings of the 15th International Conference on Agile Software Development (XP 2014),* T. Dingsøyr, N.B. Moe, R. Tonelli, S. Counsell, C. Gencel and K. Petersen (eds.), Rome, Italy: Springer, Cham, pp. 1-8.

Dingsøyr, T., Moe, N. B., Fægri, T. E., and Seim, E. A. 2018. "Exploring Software Development at the Very Large-Scale: A Revelatory Case Study and Research Agenda for Agile Method Adaptation," *Empirical Software Engineering* (23), pp. 490-520.

Dingsøyr, T., Nerur, S., Balijepally, V., and Moe, N. B. 2012. "A Decade of Agile Methodologies: Towards Explaining Agile Software Development," *Journal of Systems and Software* (85:6), pp. 1213-1221.

Dubinsky, Y., Talby, D., Hazzan, O., and Keren, A. 2005. "Agile Metrics at the Israeli Air Force," *Proceedings of the Agile Development Conference (ADC'05)*, Denver, CO, USA: IEEE, pp. 12-19.

Dumitriu, F., Meșniță, G., and Radu, L.-D. 2019. "Challenges and Solutions of Applying Large-Scale Agile at Organizational Level," *Informatica Economica* (23:3), pp. 61-71.

Ebert, C., Dumke, R., Bundschuh, M., and Schmietendorf, A. 2005. *Best Practicesin Software Measurement: How to Use Metrics to Improve Project and Process Performance*, (1 ed.). Heidelberg, Germany: Springer.

Eckerson, W. W. 2010. *Performance Dashboards: Measuring, Monitoring, and Managing Your Business*, (2 ed.). John Wiley & Sons.

Fenton, N., and Bieman, J. 2014. *Software Metrics: A Rigorous and Practical Approach*, (3 ed.). CRC Press.

Fenton, N. E., and Neil, M. 2000. "Software Metrics: Roadmap," *Proceedings of the Conference on the Future of Software Engineering*, Limerick, Ireland: ACM, pp. 357-370.

Fontana, A., and Frey, J. H. 2000. "The Interview: From Structured Questions to Negotiated Text," *Handbook of qualitative research* (2:6), pp. 645-672.

Geras, A., Smith, M., and Miller, J. 2004. "A Prototype Empirical Evaluation of Test Driven Development," *Proceedings of the 10th International Symposium on Software Metrics*, Chicago, IL, USA: IEEE, pp. 405-416.

Hamed, A. M. M., and Abushama, H. 2013. "Popular Agile Approaches in Software Development: Review and Analysis," *Proceedings of the International Conference on Computing, Electrical and Electronic Engineering (ICCEEE)*, Khartoum, Sudan: IEEE, pp. 160-166.

Hartmann, D., and Dymond, R. 2006. "Appropriate Agile Measurement: Using Metrics and Diagnostics to Deliver Business Value," *Proceedings of the AGILE 2006 (AGILE'06)*, Minneapolis, MN, USA: IEEE, pp. pp. 6 pp.-134.

Highsmith, J., and Cockburn, A. 2001. "Agile Software Development: The Business of Innovation," *Computer* (34:9), pp. 120-127.

Highsmith, J. A., and Highsmith, J. 2002. *Agile Software Development Ecosystems*. Addison-Wesley Professional.

Hossain, S. S., Ahmed, P., and Arafat, Y. 2021. "Software Process Metrics in Agile Software Development: A Systematic Mapping Study," *Proceedings of the International Conference on Computational Science and Its Applications*, Cagliari, Italy: Springer, Cham, pp. 15-26.

ISO. 2017. "Iso/Iec/Ieee International Standard - Systems and Software Engineering–Vocabulary," in: *ISO/IEC/IEEE 24765: 2017*.

Janus, A., Dumke, R., Schmietendorf, A., and Jäger, J. 2012. "The 3c Approach for Agile Quality Assurance," *3rd International Workshop on Emerging Trends in Software Metrics (WETSoM)*, Zurich, Switzerland: IEEE, pp. 9-13.

Javdani, T., Zulzalil, H., Ghani, A. A. A., Sultan, A. B. M., and Parizi, R. M. 2012. "On the Current Measurement Practices in Agile Software Development," *International Journal of Computer Science Issues (IJCSI)* (9:4), pp. 127-133.

Jethani, K. 2013. "Software Metrics for Effective Project Management," *International Journal of System Assurance Engineering and Management* (4:4), pp. 335-340.

Kaplan, R. S., and Norton, D. P. 1992. "The Balanced Scorecard: Measures That Drive Performance," *Harvard Business Review* (83:7), pp. 71-79.

Kettunen, P., Laanti, M., Fagerholm, F., and Mikkonen, T. 2019. "Agile in the Era of Digitalization: A Finnish Survey Study," *Proceedings of the International Conference on Product-Focused Software Process Improvement*, Barcelona, Spain: Springer, Cham, pp. 383-398.

Kišš, F., and Rossi, B. 2018. "Agile to Lean Software Development Transformation: A Systematic Literature Review," *Proceedings of the Federated Conference on Computer Science and Information Systems (FedCSIS)*, Poznan, Poland: IEEE, pp. 969-973.

Korpivaara, I., Tuunanen, T., and Seppänen, V. 2021. "Performance Measurement in Scaled Agile Organizations," *Proceedings of the 54th Hawaii International Conference on System Sciences (HICSS)*, Kauai, Hawaii, USA, pp. 6912-6921.

Kruchten, P. 2013. "Contextualizing Agile Software Development," *Journal of Software: Evolution and Process* (25:4), pp. 351-361.

Kumar, G., and Bhatia, P. K. 2012. "Impact of Agile Methodology on Software Development Process," *International Journal of Computer Technology and Electronics Engineering (IJCTEE)* (2:4), pp. 46-50.

Kupiainen, E., Mäntylä, M. V., and Itkonen, J. 2015. "Using Metrics in Agile and Lean Software Development – a Systematic Literature Review of Industrial Studies," *Information and Software Technology* (62), pp. 143-163.

Kurnia, R., Ferdiana, R., and Wibirama, S. 2018. "Software Metrics Classification for Agile Scrum Process: A Literature Review," *International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, Yogyakarta, Indonesia: IEEE, pp. 174-179.

Kütz, M. 2011. *Kennzahlen in der It: Werkzeuge Für Controlling Und Management*. Heidelberg: dpunkt. verlag.

Larman, C., and Vodde, B. 2016. *Large-Scale Scrum: More with Less*. Boston, MA: Addison-Wesley Professional.

Leffingwell, D. 2018. *Safe 4.5 Reference Guide: Scaled Agile Framework for Lean Enterprises*. Addison-Wesley Professional.

Liechti, O., Pasquier, J., and Reis, R. 2017. "Beyond Dashboards: On the Many Facets of Metrics and Feedback in Agile Organizations," *IEEE/ACM 10th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, Buenos Aires, Argentina: IEEE, pp. 16-22.

Maiden, N., and Jones, S. 2010. "Agile Requirements Can We Have Our Cake and Eat It Too?," *IEEE Software* (27:3), pp. 87-88.

Mikalsen, M., Stray, V., Moe, N. B., and Backer, I. 2020. "Shifting Conceptualization of Control in Agile Transformations," *Proceedings of the 21st International Conference on Agile Software Development (XP 2020),* M. Paasivaara and P. Kruchten (eds.), Copenhagen, Denmark: Springer, Cham, pp. 173-181.

Miles, M. B., Huberman, A. M., and Saldaña, J. 2020. *Qualitative Data Analysis: A Methods Sourcebook*. SAGE Publications, Inc.

Mills, E. E. 1988. "Software Metrics," Carnegie-Mellon University Pittsburgh - Software Engineering Institute.

Myers, M. D., and Newman, M. 2007. "The Qualitative Interview in Is Research: Examining the Craft," *Information and Organization* (17:1), pp. 2-26.

Neely, A. D., Adams, C., and Kennerley, M. 2002. *The Performance Prism: The Scorecard for Measuring and Managing Business Success*. Prentice Hall Financial Times London.

Nerur, S., Mahapatra, R., and Mangalaraj, G. 2005. "Challenges of Migrating to Agile Methodologies," *Communications of the ACM* (48:5), pp. 72-78.

Nord, R. L., Ozkaya, I., and Kruchten, P. 2014. "Agile in Distress: Architecture to the Rescue," *Proceedings of the 15th International Conference on Agile Software Development (XP 2014),* T. Dingsøyr, N.B. Moe, R. Tonelli, S. Counsell, C. Gencel and K. Petersen (eds.), Rome, Italy: Springer, Cham, pp. 43-57.

Olszewska, M., Heidenberg, J., Weijola, M., Mikkonen, K., and Porres, I. 2016. "Quantitatively Measuring a Large-Scale Agile Transformation," *Journal of Systems and Software* (117), pp. 258-273.

Oza, N., and Korkala, M. 2012. "Lessons Learned in Implementing Agile Software Development Metrics," *Proceedings of the UK Academy for Informaiton Systems Conference (UKAIS),* Oxford, UK, p. 38.

Paasivaara, M. 2017. "Adopting Safe to Scale Agile in a Globally Distributed Organization," *Proceedings of the 12th International Conference on Global Software Engineering (ICGSE)*, Buenos Aires, Argentina: IEEE, pp. 36-40.

Petersen, K., and Wohlin, C. 2009. "A Comparison of Issues and Advantages in Agile and Incremental Development between State of the Art and an Industrial Case," *Journal of Systems and Software* (82:9), pp. 1479-1490.

Poligadu, A., and Moloo, R. K. 2014. "An Innovative Measurement Programme for Agile Governance," *International Journal of Agile Systems and Management* (7:1), pp. 26-60.

Preiss, K., Goldman, S. L., and Nagel, R. N. 1996. *Cooperate to Compete: Building Agile Business Relationships*. Van Nostrand Reinhold.

Rajlich, V. 2006. "Changing the Paradigm of Software Engineering," *Communications of the ACM* (49:8), pp. 67-70.

Ram, P., Rodriguez, P., and Oivo, M. 2018. "Software Process Measurement and Related Challenges in Agile Software Development: A Multiple Case Study," *Proceedings of the International Conference on Product-Focused Software Process Improvement*, Wolfsburg, Germany: Springer, Cham, pp. 272-287.

Rolland, K., Dingsøyr, T., Fitzgerald, B., and Stol, K.-J. 2016. "Problematizing Agile in the Large: Alternative Assumptions for Large-Scale Agile Development," *Proceedings of the 39th International Conference on Information Systems*, Dublin, Ireland: Association for Information Systems (AIS), pp. 1-21.

Runeson, P., and Höst, M. 2009. "Guidelines for Conducting and Reporting Case Study Research in Software Engineering," *Empirical Software Engineering* (14), pp. 131-164.

Saldaña, J. 2021. *The Coding Manual for Qualitative Researchers*. London: SAGE Publications.

Schwaber, K. 2004. *Agile Project Management with Scrum*. Microsoft press.

Schwaber, K. 2018. "The Nexus Guide." Retrieved 11. November, 2021, from https://www.scrum.org/resources/nexus-guide

Schwaber, K., and Beedle, M. 2002. *Agile Software Development with Scrum*. Prentice Hall Upper Saddle River.

Schwaber, K., and Sutherland, J. 2020. "The 2020 Scrum Guide." Retrieved 11. November, 2021, from https://scrumguides.org/scrum-guide.html

Scrum.org. 2021. "Glossary of Scrum Terms." Retrieved 11. November, 2021, from https://www.scrum.org/resources/scrum-glossary

Seaman, C. B. 1999. "Qualitative Methods in Empirical Studies of Software Engineering," *IEEE Transactions on Software Engineering* (25:4), pp. 557-572.

Shirokova, S., Kislova, E., Rostova, O., Shmeleva, A., and Tolstrup, L. 2020. "Company Efficiency Improvement Using Agile Methodologies for Managing It Projects," *Proceedings of the International Scientific Conference-Digital Transformation on Manufacturing, Infrastructure and Service*, New York, NY, USA, pp. 1-10.

Staron, M., and Medig, W. 2011. "Factors Determining Long-Term Success of a Measurement Program: An Industrial Case Study," *e-Informatica Software Engineering Journal* (5:1).

Stettina, C. J., and Schoemaker, L. 2018. "Reporting in Agile Portfolio Management: Routines, Metrics and Artefacts to Maintain an Effective Oversight," *Proceedings of the 19th International Conference on Agile Software Development (XP 2018),* J. Garbajosa, X. Wang and A. Aguiar (eds.), Porto, Protugal: Springer, Cham, pp. 199-215.

Stolberg, S. 2009. "Enabling Agile Testing through Continuous Integration," *Proceedings of the Agile Conference*, Chicago, IL, USA: IEEE, pp. 369-374.

Storti, A., and Clear, T. 2020. "The Contradiction of Agile Measures: Customer as Focus, but Process as Measured?," *Proceedings of the Australasian Conference on Information Systems*, Wellington, New Zealand: Association for Information Systems (ACIS), pp. 1-12.

Talby, D., and Dubinsky, Y. 2009. "Governance of an Agile Software Project," *Proceedings of the ICSE Workshop on Software Development Governance*, Vancouver, BC, Canada: IEEE, pp. 40-45.

Talby, D., Hazzan, O., Dubinsky, Y., and Keren, A. 2006. "Reflections on Reflection in Agile Software Development," *Proceedings of the AGILE 2006 (AGILE'06)*, Minneapolis, MN, USA: IEEE, pp. 11 pp.-112.

Turetken, O., Stojanov, I., and Trienekens, J. J. 2017. "Assessing the Adoption Level of Scaled Agile Development: A Maturity Model for Scaled Agile Framework," *Journal of Software: Evolution and Process* (29:6), p. e1796.

Uludağ, Ö., Kleehaus, M., Xu, X., and Matthes, F. 2017. "Investigating the Role of Architects in Scaling Agile Frameworks," *Proceedings of the IEEE 21st International Enterprise Distributed Object Computing Conference (EDOC)*, Quebec City, QC, Canada: IEEE, pp. 123-132.

Uludağ, Ö., Philipp, P., Putta, A., Paasivaara, M., Lassenius, C., and Matthes, F. 2021. "Revealing the State-of-the-Art in Large-Scale Agile Development: A Systematic Mapping Study," in: *arXiv preprint arXiv:2007.05578*.

Van Oosterhout, M., Waarts, E., and van Hillegersberg, J. 2006. "Change Factors Requiring Agility and Implications for It," *European Journal of Information Systems* (15:2), pp. 132-145.