DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Information Systems

Goals and KPIs in Large-scale Agile Development: A Systematic Literature Review

Nadine Teiber

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Information Systems

Goals and KPIs in Large-scale Agile Development: A Systematic Literature Review

Ziele und KPIs in der skalierten agilen Softwareentwicklung: Eine systematische Literaturrecherche

Author:	Nadine Teiber
Supervisor:	Prof. Dr. Florian Matthes
Advisor:	Pascal Philipp, M. Sc.
Submission Date:	17.05.21

I confirm that this master's thesis in information systems is my own work and I have documented all sources and material used.

Munich, 17.05.21 Nadine Teiber

Acknowledgments

First, I would like to thank my advisor Pascap Philipp for providing support, guidance and constructive feedback during the thesis.

Furthermore, I would like to thank my supervisor Prof. Dr. Florian Matthes for giving me the opportunity of carrying out my master's thesis at the chair for Software Engineering for Business Information Systems (SEBIS).

Moreover, I would like to thank my family and my friends for their support during this journey.

Abstract

Although agile software development was originally designed for use in small, single-team projects [1], the adoption of agile software development in large-scale projects has increased over the last years [2]. Within the adoption of large-scale agile development, coordination in multi-team environments, hierarchical management and organizational boundaries has proven to be challenging [3]. With implementing agile at large organizations, also comes the need to introduce software measurements into the development process. Software measurement has been widely studied in traditional software development [4], and over the last years also some literature reviews about agile and lean software development have been performed [5], [6]. However, in a systematic mapping study about large-scale agile software development, Uludat et al. [7] identified that so far, there is no research stream about measurements in large-scale agile software development.
Since no systematic literature review about metrics in large-scale agile software development has been performed, this master's thesis aims to close this research gap. A systematic literature review was conducted to identify goals and metrics in large-scale agile software development as well as the challenges and success factors of introducing and using metrics. With the search terms, 17042 publications were found. Out of those, 110 primary studies about goals and metrics in large-scale agile software development were identified.
More than 80% of studies were published in conferences, and 33% of sources were experience reports, indicating a lack of maturity. Furthermore, we found that most metrics are measured on team and program levels. Even though multiple studies indicate the use of metrics at the portfolio level, a lack of actually described metrics has been found. Furthermore, 21 challenges of using and implementing metrics in large-scale agile software development have been identified that could be grouped into six categories. The most frequently mentioned challenges were problems with data availability and data collection. Moreover, 22 success factors grouped into ten categories were found. The most prominent success factors were automated data collection, making metrics visible to everyone, and iteratively introducing metrics.

Contents

1. Introduction

This chapter starts by describing the motivation of this master's thesis. Then the objectives and research questions are described. Section 1.3 then describes the research approach that was chosen to answer the research questions.

1.1. Motivation

Agile software development was introduced with the Agile Manifesto published in 2001 [8] and was developed to adapt to the volatile requirements of today's economy. Over the last years, agile software development methods like Scrum, Extreme Programming (XP), or Kanban has become the most used software development methods [9]. Originally, agile software development was designed for small, single-team projects [1]. Nonetheless, the adoption of agile software development in large-scale projects has increased over the last years [2]. Within the adoption of large-scale agile development, coordination in multi-team environments, hierarchical management, and organizational boundaries has proven to be challenging [3].
Software measurement has been widely studied in traditional, plan-based software development [4]. Contrary to traditional software development approaches, agile software development seeks value in delivering working software as the primary measure of progress [8]. In a Scientific Literature Review (SLR) about the use of metrics in agile software development Kupiainen et al. [5] identified that metrics are useful for sprint and project planning, progress tracking, understanding and improving quality, fixing software process problems, and motivating people. With implementing agile at large organizations, also comes the need to introduce software measurements into the development process. In large organizations, agile portfolio management is used to connect, and the strategy of operations with the portfolio of products [10]. The quality of portfolio management can be influenced by program and project reporting [11]. However, in a systematic mapping study about large-scale agile software development, Uludat et al. [7] identified that no literature reviews about measurements in large-scale agile software development have been conducted so far.
While measurements are effective for planning, progress tracking, or quality checking, there are disadvantages of metrics that can occur when measurements are not aligned right or misused [5]. Furthermore, the Version One 14th Annual Survey [12] found that fragmented tooling and project-related data as a general challenge of large-scale agile transformations.
This master's thesis is aimed to close this research gap and provide an overview of goals and metrics in large-scale agile software development by performing a SLR. Furthermore, it aims to provide an overview of the challenges and success factors of metric implementation and

usage.

1.2. Research questions

This master thesis aims to provide an overview of goals and metrics used in large-scale agile software development. In particular, we are interested in providing an overview of metrics and goals, and the challenges and success factors of metric usage and implementation. To analyze those aspects, three research questions have been defined, which are described in the following. Research questions 2 and 3 have been defined by following the example of Dikert et al. [3].

> **Research Question 1:** *What are goals and metrics in large-scale agile software development?*

To answer this research question, multiple mappings have been conducted. First, metrics were extracted from the primary studies. Equal metrics were combined and their frequency of occurrence was calculated. To then provide a better overview about the metrics that are used, a classification of the metrics according to Fenton and Pfleeger [13] was performed. Furthermore, the hierarchical level at which the measurement was conducted was identified for each metric. Furthermore, goals that were mentioned in primary studies were extracted.

> **Research Question 2:** *What challenges of using and implementing metrics in large-scale agile software development have been reported?*

To answer the second research question, challenges related to the use and introduction of metrics in large-scale agile software development have been extracted. The extracted challenges were then analyzed and clustered into categories. Furthermore, the frequency of the challenge categories and their subcategories was calculated and visualized in a table.

> **Research Question 3:** *What success factors of using and implementing metrics in large-scale agile software development have been reported?*

Regarding the third research question, the approach was very similar to the method applied to answer research question three. Success factors were extracted from primary studies, analyzed, categorized, and their frequency of occurrence was calculated.

1.3. Research approach

This section provides an overview of the research approach applied in this master's thesis. To answer the research questions, a SLR approach was chosen. This approach was selected because SLRs can be used to summarize existing evidence, identify gaps in current literature and provide a background for new research activities [14]. Accordingly, this SLR aims to give an overview of the goals and metrics used in large-scale agile software development,

as well as challenges and success factors of metric usage or implementation. Furthermore, gaps in current literature regarding the use of KPIs in large-scale agile software development shall be identified to provide an overview of future work. This SLR follows the guidelines described by Kitchenham and Charters [15] and Zhang et al. [16] and is visualized in Figure 1.1. According to Kitchenham and Charters [15] there are three main phases when conducting a SLR: planning the review, conducting the review, and reporting the review. During the planning phase, the need for a review is identified, the research questions are defined, and a search strategy is defined. In the review conduction phase, the search is performed, the primary studies are selected due to inclusion and exclusion criteria, and data is extracted. Then, the extracted data is synthesized. In the reporting phase, the results of the literature review are presented. [15] In the following, the identification of the search strategy, the identification of sources, and the data extraction process are described.

Identification and filtering of sources

In order to identify relevant publications about goals and metrics in large-scale agile software development, a search strategy that maps to the research question defined in Section 1.2 has to be defined. The search process started with deriving inclusion and exclusion criteria from the research objectives. Then the search strategy was defined by following the example of Zhang et al. [16]. The search process in this thesis consisted of two parts: the preliminary search and the main search. By performing both a manual and an automated search within the preliminary search, two objectives were aimed to be achieved. First, it should be assured that no previously known relevant sources were lost. Second, the quality of the search strings should be assessed and refined. This was done by using the preliminary search results to assess the completeness of the automated search results. The automated main search was performed on six databases (see Table 4.1). After the preliminary and main search was performed, the results were merged. The filtering process started with the removal of duplicates of publications between the two searches and the different databases. Afterward, the inclusion and exclusion criteria were used to identify relevant sources by filtering by meta-data, abstract and full-text. The full filtering process resulted in the identification of 110 primary studies.

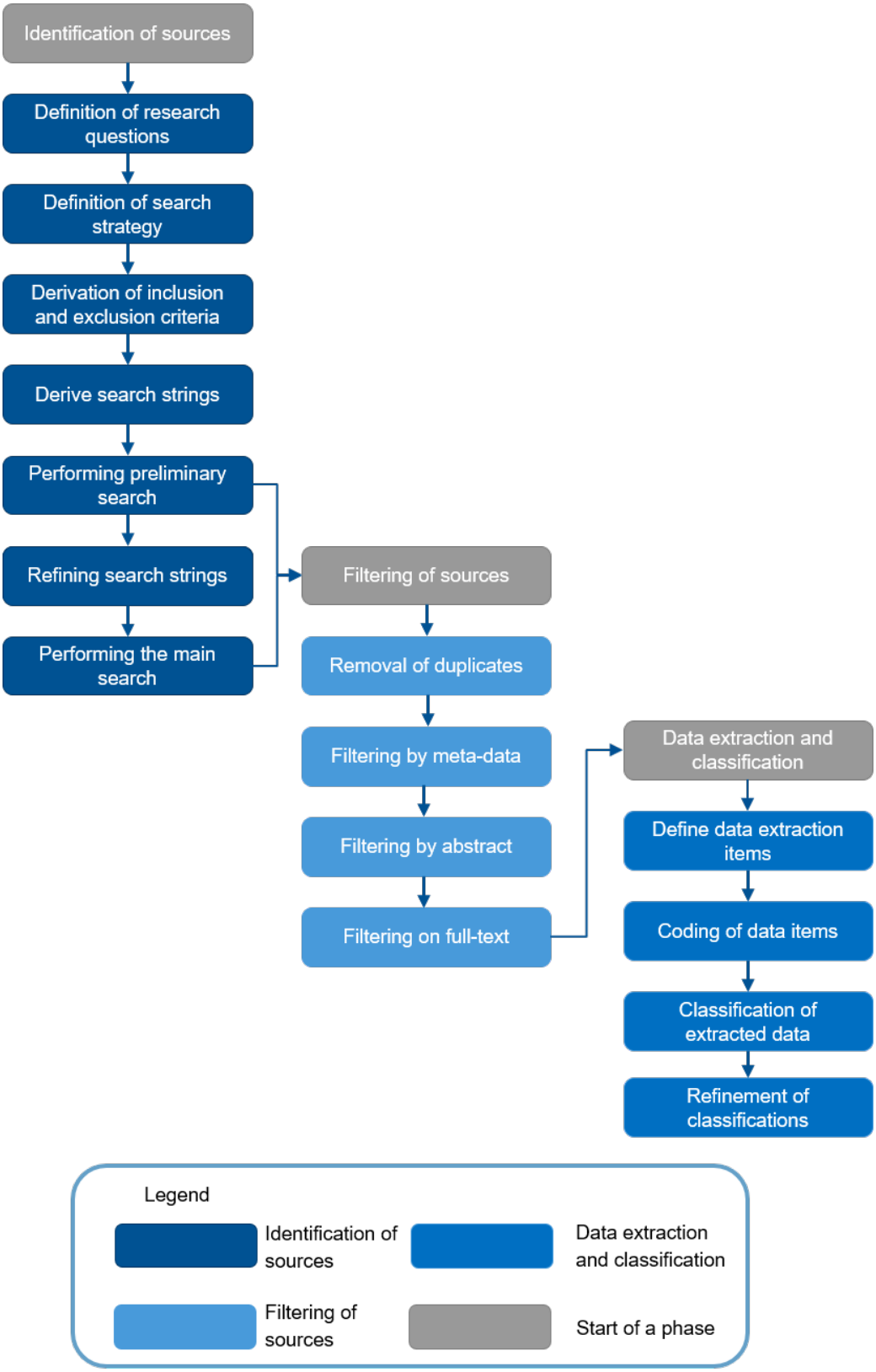

Figure 1.1.: Overview of the research approach

Data extraction and classification

After the 110 primary studies were identified, the data extraction process started. During the data extraction process, 13 categories were extracted (see Table 4.2) into an Excel sheet. In order to enhance the quality of the extracted data and classify the results, multiple iterations of this process were performed. After all criteria were extracted, the findings were synthesized into high-level categories. For the classification of criteria C1-C6, C8, and C10, existing classification schemes were used. For the remaining data extraction categories, the classification was performed in multiple iterations. The results were presented to a second researcher in regular intervals, who then gave feedback about the performed mapping. After all the conflicts were resolved, the data extraction process was completed.

Structure of the thesis

The remainder of this master's thesis is structured as follows. Chapter 2 describes the theoretical foundations of agile software development, large-scale agile development and the theory behind software measurement. Chapter 3 gives an overview of related work in software measurement in general as well as software measurement in agile software development. In Chapter 4 the research methodology is thoroughly described. Chapter 5 describes the findings of the SLR. In Chapter 6 the key findings and limitations are discussed and Chapter 7 concludes this master's thesis with a summary and an outlook for future work.

2. Foundations

2.1. Agile software development

Agile software development has been introduced in contrast to the traditional, plan-based development methods as waterfall development. Highsmith and Cockburn [17] defined it in the following way:

> *"Agility, ultimately, is about creating and responding to change. What is new about agile methods is not the practices they use but their recognition of people as the primary drivers of project success, coupled with an intense focus on effectiveness and maneuverability. This yields a new combination of values and principles that define an "agile" world view."*

To better understand the context of agile software development, the agile manifesto and the most often identified frameworks in the primary sources will be described.

2.1.1. The agile manifesto

Agile software development was founded with the agile manifesto in February 2001. The Manifesto for Agile Software Development was created by 17 individuals, consisting of representatives of extreme programming, Scrum, Crystal, Feature-Driven Development, and other agile frameworks [8]. To discover better ways of building software, Beck et al. [8] defined four core values of agile software development:

- "Individuals and interactions over processes and tools": The focus lies on close team relationships.
- "Working software over comprehensive documentation": The team aims to deliver working software in regular intervals while focusing on keeping the code simple and only creating necessary documentation.
- "Customer collaboration over contract negotiation": Customer, users, management, and teams should work together and collaborate to achieve a shared understanding and respond to change instead of only focusing on the contract.
- "Responding to change over following a plan": Introducing a plan is good to achieve an overview of a project but plans change, and those changes should be welcomed.

It is important to note that documentation, processes, contracts, and plans are still recognized as important artifacts. However, the focus should lie on individuals, working software,

collaboration, and responding to change [18]. Apart from the four core values, Beck et al. [8] also defined 12 principles of agile software development that can be found in Table 2.1. The aim of agile software development, according to Cockburn and Williams [19] is embracing change.

ID	Principle
1.	Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2.	Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3.	Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4.	Business people and developers must work together daily throughout the project.
5.	Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6.	The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7.	Working software is the primary measure of progress.
8.	Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9.	Continuous attention to technical excellence and good design enhances agility.
10.	Simplicity–the art of maximizing the amount of work not done–is essential.
11.	The best architectures, requirements, and designs emerge from self-organizing teams.
12.	At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Table 2.1.: Principles behind the Agile Manifesto by Beck et al. [8]

2.1.2. Extreme Programming

Extreme programming has been developed to overcome the problems of the "traditional", plan-based software development methods, and its practices are by no means new [20]. Rather extreme programming is a combination of multiple practices that have been defined in multiple frameworks so far [20], forming a set of practices and a process that needs to be followed. Those practices are described in Table 2.2. As described by Beck [21], the extreme programming process is comprised of six steps: exploration, planning, iterations to release, productionizing, maintenance, and death. In the exploration phase, user stories that should be included in the release are created. Furthermore, developers familiarize themselves with the technology and the architecture of the technology and practices. During the planning phase, the user stories that should be developed in the next small release are estimated and prioritized. After planning, multiple iterations are performed to deliver the release. In the

first iteration, the system architecture is built to be able to deliver a working system after each iteration. At the end of each iteration, which should take between one to four weeks, functional tests created by the customer are run. In the productionizing phase, the release is tested by the customer and wanted changes to the system are documented. After the system is released to the customer, the project enters the maintenance phase. The product has to be kept running while new releases are developed. After all customer requirements are developed, and all releases are integrated, the final documentation is written and handed off to the customer. [21]

Practice	Description
Planning game	There is a collaboration between customers and developers. While the developers estimate the user stories, customers use those to decide about the scope and the timing of releases.
Small releases	The system is developed within a few months, and new releases are frequently produced
Metaphor	The system is defined by a metaphor or a set of metaphors that are shared between customer and developer.
Simple design	The aim is to design the most straightforward solution possible, therefore reducing duplicated code and complexity.
Tests	Test-driven software development is used. Unit tests that developers write, and functional tests written by customers, are executed continuously.
Refactoring	The system is constantly being refactored by removing duplicates, simplifying, and improving communication
Pair programming	All code is written by two people at one computer
Continuous integration	New code is continuously integrated with the current system. When the system is built, all existing tests must pass. Otherwise, all changes will be discarded.
Collective ownership	Anyone can change and improve every part of the system at any time
On-site customer	The customer is present at the team site at all times.
40-hour week	Two consecutive weeks of overtime should not occur and are treated as a deeper problem in the process that should be addressed.
Open workspace	Teams work in a large room with small cubicles. Pair programming spaces are set up in the center of the room.
Just rules	Each team member has to adhere to the same rules. But teams can decide to change the rules if a shared agreement has been reached and the effects of the change were assessed

Table 2.2.: Practices of extreme programming as defined by Beck [20]

2.1.3. Scrum

The term Scrum was first mentioned by Takeuchi and Nonaka [22] who described a new product development process. Instead of the old, sequential approaches, they argued that Scrum, "a holistic or "rugby" approach-where a team tries to go the distance as a unit, passing the ball back and forth may better serve today's competitive requirements" [22]. Scrum is a framework that describes a set of best practices collected from industry, not just a simple development process with predefined steps [23]. Schwaber [24] divides the Scrum process into three phases: pregame, development, and postgame. During the pregame phase, two steps are performed. The release is planned by estimating the schedule and the system architecture is designed. During the game phase the actual development of new functionality is performed in iterative sprints. In the postgame phase the product is then prepared for release by finishing the documentation and final testing. [24]

Roles

Within Scrum, the management tasks are divided into three roles: the Scrum Master, the Product Owner, and the team [25]. The Product Owner represents the interests of the stakeholders in the project and is responsible for creating requirements in cooperation with the customer. He is responsible for prioritizing the product backlog, a list of all requirements, to ensure that the most valuable requirements are developed first. The teams' task is to develop functionality in a self-organized manner. They manage the development of the backlog items within an iteration to deliver working software. The Scrum Masters task is to remove obstacles in the implementation of the Scrum process by teaching the Scrum to project members and aligning it with the organization. [25]

Artifacts

Three new artifacts are described in the scrum process: the product backlog, the sprint backlog, and the burndown chart [25]. The product backlog contains all requirements of a product. These requirements are put into the backlog by the Product Owner in the form of Product Backlog Item (PBI)s. For each PBI, the initial time estimate, adjusted estimate, and the remaining work are recorded. The Product Backlog is not fixed. Rather it evolves throughout the project as the product evolves. [25]
The amount of remaining work across time within a product is visualized with a burndown chart [25]. Within a burndown chart, the remaining workdays are visualized on the y-axis, while the x-axis shows a timeline. The resulting trend line allows predicting the influence of added or changed PBIs in terms of the predicted completion of the project. [25]
The Sprint Backlog includes all PBIs that a team agreed to implement of the product backlog and is used to present a picture of the work that could be accomplished during a sprint [25]. These tasks are decided on and estimated in the sprint planning meeting. Within the Sprint Backlog, tasks should not be estimated to take longer than 4 to 16 hours and can only be changed by the team itself [25].

Process

The development process, as visualized in Figure 2.1, in Scrum is performed in sprints. At the beginning of each sprint, which usually has a duration of 30 days, a sprint planning meeting is performed. In this meeting, the Product Owner and the team members discuss what will be implemented in the next iteration. The Product Owner hands the team the prioritized product backlog, and the team decides how much effort they can deliver. All backlog items that are estimated and agreed to be developed in the next sprint are put into the sprint backlog. During the sprint, a daily Scrum meeting is held by the team. Here each team member describes what he has done, what he is planning on doing, and what impediments have occurred since the last daily Scrum meeting. The sprint review meeting takes place at the end of each sprint. Here, the team presents the developed sprint backlog items to the Product Owner and all other interested stakeholders. After the sprint review, the Scrum Master performs the sprint retrospective with the team to identify issues with the Scrum process and help the team to increase productivity and satisfaction. [25]

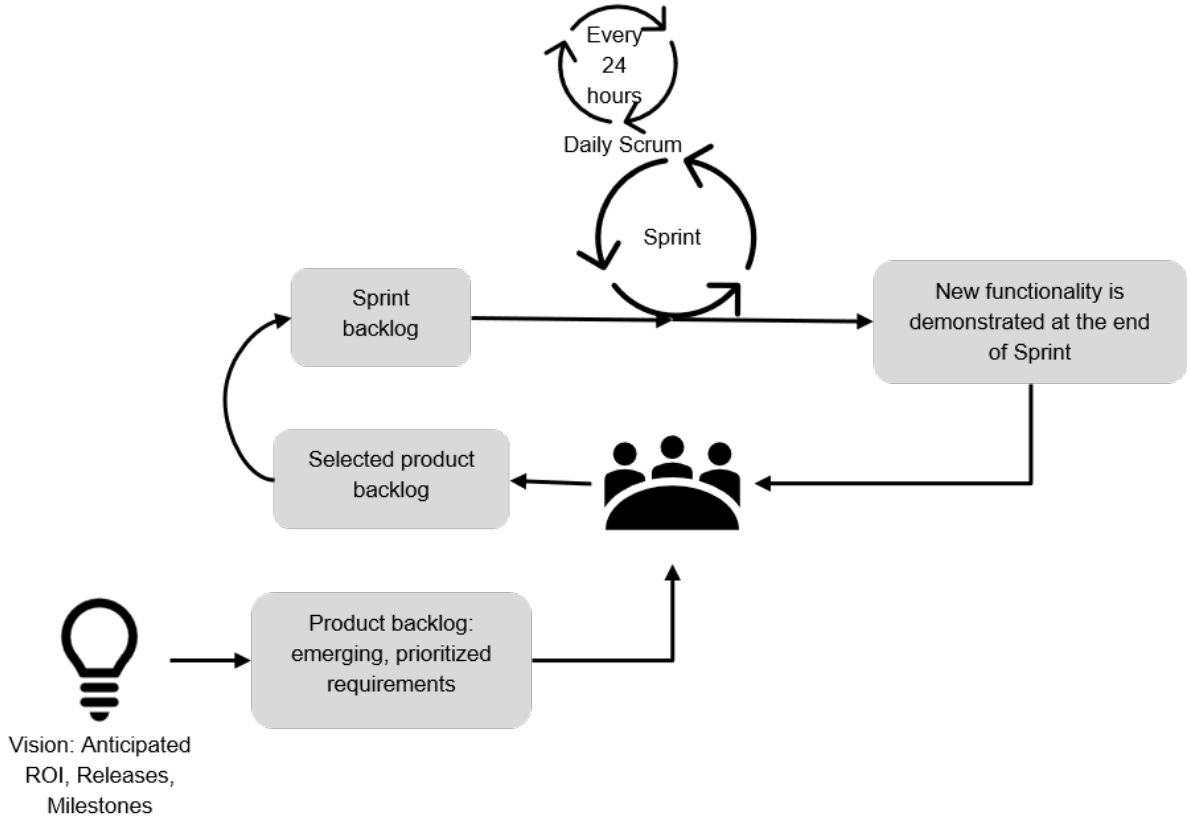

Figure 2.1.: Scrum process overview by Schwaber [25]

2.1.4. Lean software development

The concept of "lean" was originally developed as an approach for the manufacturing industry [26] and was later applied to the software development context by Poppendieck and Poppendieck [27]. The seven Lean software development principles as described by Poppendieck and Poppendieck [27] are:

1. Eliminate waste. Everything that does not create value for the customer is considered waste and should be avoided.
2. Amplify learning. In order to continuously improve a software development environment, learning is necessary.
3. Decide as late as possible. This is necessary when decisions are made in the uncertain context of the economic market.
4. Deliver as fast as possible. Developing in shorter cycles increases customer feedback and gives the possibility to learn how to optimize the development process.
5. Empower the team. In Lean software development, teams work autonomously by using a pull mechanism of work items that are developed.
6. Build integrity in. Software should be built with a focus on perceived integrity, conceptual integrity, and usefulness over time.
7. See the whole. It is important to try to maximize the overall performance. Therefore people should not be measured on their specialized contribution to avoid sub-optimization.

Lean software development coincides very well with the principles stated in the agile manifesto as described in Table 2.1. The only difference that can be found between agile and Lean is the focus on the seventh principle mentioned above: "See the whole" [27, 6].

2.2. Large Scale Agile Software Development

The agile software development methods described above were originally designed for small projects performed by a single team [1]. Adopting agile software development in large-scale settings has been proven challenging due to coordination challenges between teams, differing approaches in multi-team settings, and hierarchical settings [3].

Definition of large-scale development

In the literature, there exist multiple definitions about when a project can be considered as large-scale [28]. These definitions can be separated into four categories:

1. Number of people working on the project is used as an interpretation of large-scale in many studies, with the number of people varying between 40 [29] and 300 people [30].

2. Number of teams as a definition of large-scale also varies very much in size. Dikert et al. [3] defined large-scale as at least six teams working on a project [3], while Dingsøyr et al. [31] considered a project with two or more teams as large-scale.

3. Project size. A project was considered large-scale when it consists of more than 5 million lines of code [32], or 60-80 features are developed in the scope of two years [33].

4. Project cost of more than 10 million GBP in combination with more than 50 team members was used as a definition in a study of Berger et al. [34].

To develop a shared understanding Dingsøyr et al. [31] proposed a taxonomy of scale for agile software development projects. As visible in Table 2.3, they divided agile projects into three levels. When only one team is involved in the project, it is considered small-scale. Large-scale projects have two to nine teams working on them, and when ten or more teams are working on a project, it is considered very large-scale [31]. In combining the definitions of Dingsøyr et al. [31], and Dikert et al. [3] we consider three or more collaborating teams or more than 50 people working together on a project as large-scale. Furthermore, the application of agile practices in companies as a whole [35] is also considered to be large-scale agile.

Level	Number of teams	Coordination approaches
Small-scale	1	Coordinating the team can be done using agile practices such as daily meetings, common planning, review and retrospective meetings.
Large-scale	2-9	Coordination of teams can be achieved in a new forum such as a Scrum of Scrums forum.
Very large-scale	10+	Several forums are needed for coordination, such as multiple Scrum of Scrums.

Table 2.3.: Taxonomy of scale of agile software development projects by Dingsøyr et al. [31]

2.3. Software measurement

"You can not control what you can not measure" (De Marco, 1982)

Software measurement has been essential for understanding and evaluating the software development process and the resulting products in order to track and improve their performance [36]. Software measurement can be used for two different use cases: prediction and assessment [37]. Predictive measurement can be used to achieve accurate estimations by using past project data [37] or predict system reliability [38]. Measurement is the process of assigning values to attributes or entities according to predefined rules [13]. Fenton and Pfleeger [13] subdivide entities into three different parts: product, process, and resources.

Attributes are measurable features or characteristics of the entities, like project cost or the number of product defects. Each measure must be linked to a specific and understandable measurement objective [13].

Software metrics have been widely studied and multiple literature reviews have been conducted, which are further described in Chapter 3.

2.3.1. Measurement programs

When introducing measurement programs, software development organizations face the problems of collecting too much or the wrong data or performing the wrong measurements [39]. To overcome this challenge, multiple software measurement approaches have been described in scientific literature. The most used measurement program in scientific research GQM [4], and the GQM+ model will be described in the following section.

Goal-Question-Metric approach

Basili and Weiss [40] first introduced the Goal Question Metric (GQM) approach, which is used to integrate goals into the measurement process to achieve purposeful measurement. The GQM approach consists of three levels [41]:

1. Goal: On the conceptual level, a goal is defined for one of the three possible entities products, processes, and resources.
2. Question: On the operational level, questions are derived from the goal that must be answered to determine whether the goals have been met.
3. Metric: On the qualitative level, measures are defined to answer the questions identified in the previous step.

As visualized in Figure 2.2, in the GQM multiple questions can be linked to one goal. Each question is then linked to one or multiple metrics. Furthermore, one metric may be used to answer multiple questions that are linked to multiple goals [41].

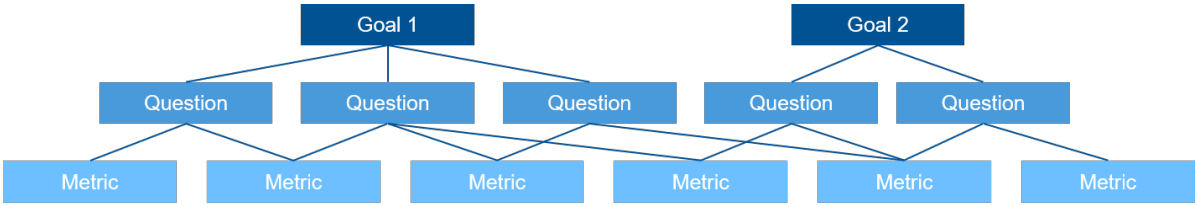

Figure 2.2.: GQM Approach by Basili et al. [41]

GQM+Strategies

In order to provide transparency of measurement goals at different organizational levels, Basili et al. [39] have extended the GQM by defining the GQM+Strategies model. An overview

of the GQM+Strategies by Basili et al. [39] is provided in Figure 2.3. The process starts at the business level by defining a business goal and an corresponding strategy for achieving this goal. Based on those strategic elements, the GQM approach described in the previous section is applied. First, a measurement goal is defined. Then a set of questions is defined that can be answered by the metrics. After all goals, questions, and metrics on the business level are defined, the process is repeated at the software level and afterward at the project level. The only difference when applying the process at the lower levels is that the software or project goals are derived from the strategy and defined in the upper levels. [39] Basili et al. [39] claim that introducing GQM+Strategies into the measurement programs results in increased transparency of measurement goals at different organizational levels, improves communication across levels, reduces the cost of measurement, and helps in building an experience base.

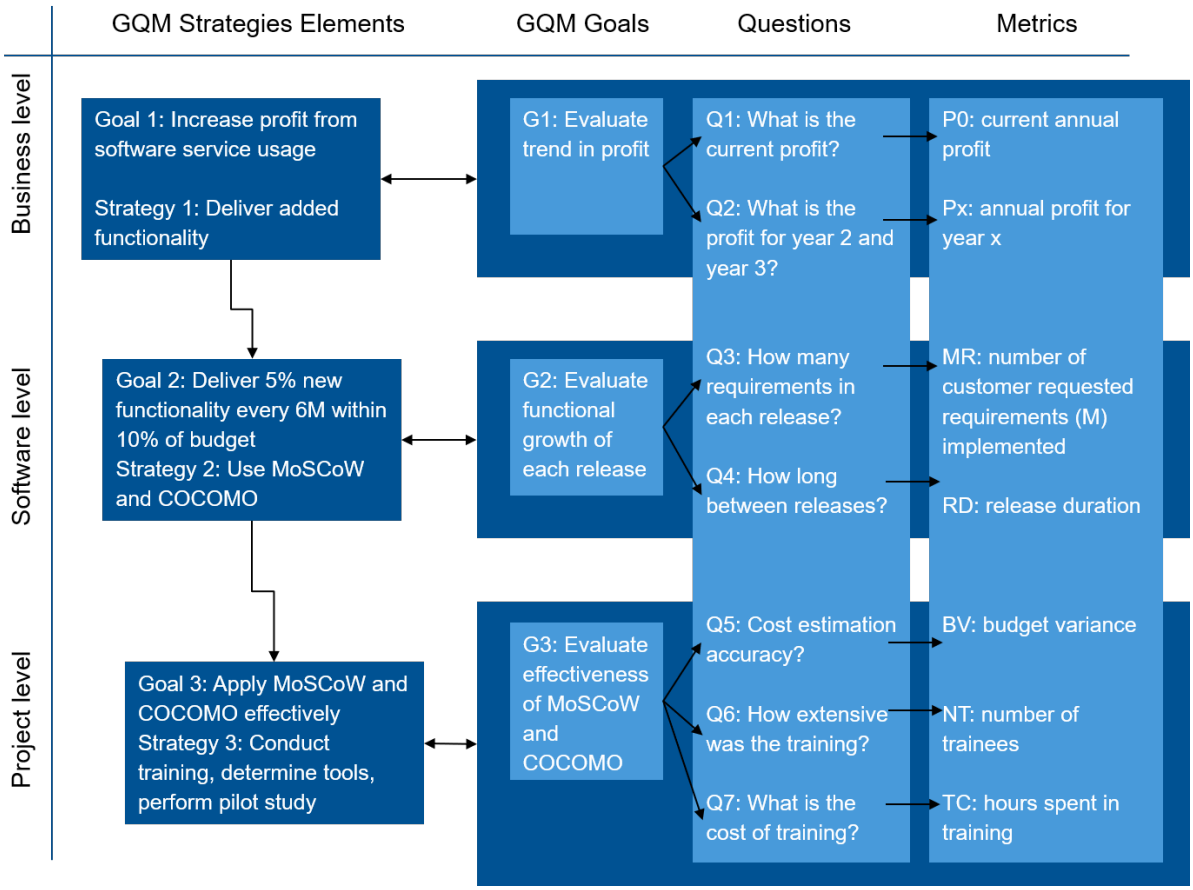

Figure 2.3.: GQM+ Strategy by Basili et al.[39]

3. Related Work

This chapter describes related work about systematic literature reviews and systematic mapping studies performed on the topic of measurement in traditional and agile software development. The studies mentioned in this section were identified in the preliminary and main search described in Chapter 4. This Chapter extends the foundations in Chapter 2. First, related work in software development in general is presented, followed by literature reviews and Systematic Mapping Study (SMS) performed on software measurement in agile and Lean software development.

3.1. Software measurement

Software metrics have been widely studied and multiple literature reviews have been performed so far in the traditional software development context [42].

Meidan et al. [4]

Meidan et al. [4] performed an extensive systematic mapping study to understand the measurement of the software development process. Following the recommendations of Kitchenham and Charters [15], they identified 462 primary studies. By analyzing publication trends, Meidan et al. [4] identified an increasing interest in the research streams of monitoring the development process to control and manage it, assessment and improvement, and measurement concepts and practices. All measurements found in the literature were classified into five different levels of abstraction: organization, process, process model, project, and developer. Those abstraction levels were used to describe at which level the measurement is conducted. Although project and process were found to be the most measured entities, there is an increasing amount of measurements conducted at individual and team levels. Furthermore, new attributes like team-climate and creativity have emerged. In their 462 primary studies, Meidan et al. [4], were able to identify 64 different measurement attributes [13], out of which productivity was found to be the most measured one. Within the primary studies, GQM and Capability Maturity Model Integration (CMMI) were found to be the most used models. [4]

Kitchenham [42]

To identify trends in influential publications about software metrics, Kitchenham [42] performed a preliminary mapping study. Using Scopus Kitchenham [42] identified the 25 most cited papers between 2000 and 2005. Out of those studies, the research type, goal, and topic

were extracted. Fault prediction (12 studies), re-engineering, error-prediction (10 studies), and component-fault classification (8 studies) were the most cited goals stated in the primary studies. Kitchenham [42] concluded that the most influential papers in software development are journal publications, where data is evaluated empirically. [42]

Gómez et al. [43]

Gómez et al. [43] performed a systematic literature review to answer the questions of how to measure, what to measure, and when to measure. By performing an automated search in the databases ScienceDirect, ACM, IEEE, and Wiley, the authors identified 78 publications. To answer the question of what to measure, the entities and attributes of the measures were analyzed. The identified measures were grouped into the three entities process (9 percent), project (12 percent), and product (79 percent). Complexity (19 percent), size (16 percent), inheritance (8 percent), defects (7 percent), and structuredness (7 percent) were the most measured attributes. Gómez et al. [43] furthermore classified the measures to the waterfall life-cycle model to identify when measures are taken. They discovered that most measures are used during the design phase (42 percent), followed by development (27 percent), maintenance (14 percent), testing (12 percent), and analysis (5 percent). [43]

3.2. Software measurement in agile software development

Kupiainen et al. [5]

Kupiainen et al. [5] present an SLR on metrics used in industrial agile software development. By analyzing 30 publications, they provide an overview of reasons for using metrics and their effects. For the implementation of the SLR, the recommendations of Kitchenham et al. [44] were followed. The automated search was performed in Scopus. Overall, Kupiainen et al. [5] were able to identify 102 metrics that could be grouped into five reasons for usage: sprint and project planning, sprint and project progress tracking, understanding and improving quality, fixing software process problems, and motivating people. Furthermore, they implemented a mapping between the metrics and the agile software development principles categorized by Patel et al. [45]. Effort estimate, defect count, and velocity were identified as high influence metrics due to their number of occurrences. Customer satisfaction, technical debt, build status, and progress as working code was identified as highly relevant in the primary studies. Since they were not measured frequently, Kupiainen et al. [5] stated that those highly relevant metrics should be investigated further. [5]

Feyh et al. [6]

Feyh et al. [6] performed a systematic mapping study to identify metrics in the context of Lean software development. The study's methodology followed the guidelines of Petersen et al. [46] and Kitchenham and Charters [15]. In their mapping study, they were able to identify 27 publications dealing with metrics in lean software development or Kanban context. Feyh et

al. [6] applied the ISO/IEC 15939 measurement information model for measurement program planning [47] to group the measures into base measures, derived measures, and indicators. Base measures quantify attributes, derived measures comprise multiple base measures, and indicators are defined as measures that fulfill an information need [47]. Indicators can combine multiple base measures and derived measures [47]. Overall, Feyh et al. [6] identified 22 base measures, 13 derived measures, and 14 indicators used in Lean software development or Kanban. To give an overview of the reasons for the use of measures, they performed a mapping between the measures and the seven lean software development principles as defined by Poppendieck et al. [48]. Those principles are eliminating waste, building in quality, creating knowledge, defer commitment, delivering fast, respecting people, and optimizing the whole [27]. They identified a lack of measures for respecting people, creating knowledge, and deferring commitment [6].

Ramirez et al. [49]

Ramirez et al. [49] performed a SMS to analyze how productivity is defined and measured in the context of agile software development. They followed the research methodology defined by Kitchenham et al. [50]. The metrics found in the 25 identified primary sources were grouped into three categories: productivity metrics (including quality), productivity metrics not including quality, and no metrics used. In their study, Ramirez et al. [49] found that less than fifty percent of their studies are using metrics to measure performance. Furthermore, it was found that performance is mostly measured by looking at effort spend for development or the amount of delivered software. Quality measures were related to the number of defects and the meantime to fix them. [49]

Shah et al. [51]

Another publication about productivity in agile software development was performed by Shah et al. [51]. By performing a scoping study approach developed by Arksey and O'Malley [52] they identified a gap in research regarding productivity measures used in agile software development. Within 124 studies about productivity in agile software development, they could only identify 12 studies where productivity is measured. Furthermore, Shah et al. [51] argue that knowledge should be integrated with agile productivity metrics as agile software development requires different competencies to be aggregated in a team effort. [51]

Kišš et al. [53]

Kišš et al. [53] performed a SLR to identify benefits, challenges, and metrics of agile-to-lean transformations. The research was conducted after the guidelines developed by Kitchenham and Charter [15]. Within their 18 primary sources, they were able to identify that lead time, defect count, fix-time for defects, velocity, lines of code, story rate per iteration, and release frequency are used to measure the performance of an agile-to-lean transformation. [53]

Wnuk and Maddila [54]

In their systematic mapping literature review, Wnuk and Maddila [54] identified metrics in agile and lean software development that are associated with requirements engineering. They mapped each metric to one of two external attributes: quality or time. Quality metrics were used to improve requirements, and process quality or cost and time-related metrics are related to throughput or requirements flow. In their study, Wnuk and Maddila [54] found a lack of metrics measured in the industry, that are not only proposed by research. Furthermore, no metrics covering stakeholder or human aspects in requirements engineering were identified. Therefore, Wnuk and Maddila [54] concluded that there is a need for more research towards measuring requirements activities in agile and lean software development. [54]

Almeida et al. [55]

Almeida et al. [55] performed a systematic literature review to identify metrics for estimating the effort and cost of software to apply them to the Brazilian public sector. They were able to identify six metrics for estimating effort and cost within their 69 primary sources. The complexity metric story points was the most common, followed by the functional metrics function points, use case points, and COSMIC. The complexity measure velocity and the size measure lines of code occurred the least often. [55]

Kurnia et al. [56]

The literature study of Kurnia et al. [56] discusses software metrics and their role in the Scrum software development process by analyzing 13 primary sources. In connecting the identified metrics to the scrum events sprint planning, daily scrum meeting, sprint review, and sprint retrospective, they identified metrics for each scrum event. Furthermore, Kurnia et al. [56] distinguished between the four measurement perspectives team, project, management, and client. The team reliability measure velocity and story points, the measure of project, are recommended for the sprint planning event. During daily sprint meetings, sprint and release burndown charts can be used to monitor project progress. In the sprint review, customer satisfaction and business value delivered are the leading indicators of success. Project success from a management perspective is measured by earned value management during the sprint retrospective, while a job satisfaction survey is performed on the team level. [56]

Mishra and Abdalhamid [57]

In a systematic mapping study, Mishra and Abdalhamid [57] reviewed 53 publications to identify quality attributes and measures in Scrum-based software development. Their study found process quality to be the most examined issue (45 percent), followed by project quality and product quality with each 25 percent. Like employee satisfaction and motivation, people-related quality issues were only investigated in four percent of the publications and represent a gap in research. This is especially important due to the focus of agile software development on individuals instead of processes and tools. [57]

3.3. Conclusion over related work

While all nine publications described in Section 3.2 provide information about metrics in the context of agile software development, the SLR of Kupiainen et al. [5] presents the only study that investigates metrics usage over multiple agile frameworks, the whole development context, and multiple measurement reasons. Feyh et al. [6] investigated metrics only in Lean, and Kanban software development practices, and Kurnia et al. [56] and Mishra and Abdalhamid [57] in the Scrum process. Ramirez et al [49] and Shah et al. [51] only considered productivity measures, and Wnuk and Maddila [54] and Almeida et al. [55] extract measurements related to requirements engineering. To our knowledge, no SLR about metrics in large-scale agile software development has been published so far.

4. Methodology

This Chapter describes the methodology that was chosen for conducting this master's thesis. The steps to identify the relevant literature and extract the data are explained. This literature review is performed according to the guidelines for performing systematic literature reviews [15], and the recommendations of Zhang et al. [16].

4.1. Identification and filtering of sources

4.1.1. Definition of the search strategy

After the definition of research questions, explained in Chapter 1, the search strategy used to answer the questions has to be defined [14]. For the definition of the search strategy, the recommendations of Zhang et al. [16] were used. To design an effective search strategy, Zhang et al [16]. propose the following four steps:

1. Which search approach should be used: automated or manual?
2. Which databases or venues should be searched and which part of the article should be searched through search engines?
3. Which study types should be searched and what search strings should be used?
4. What time span is included in the search?

Search approach

Following Zhang et al. [16], the search approach chosen consisted of a mixture of manual and automated search. First, a preliminary search was performed, which consisted of a manual and an automated search. This was followed by the main, automated search. There were two main reasons for using a preliminary search: (i) assuring that no papers from highly relevant sources are missing by performing a manual search and (ii) assessing the search strings' quality by performing an automated search [16]. Similar to the approach of Uludag et al. [7] the manual search was comprised of two steps: (i) search through all proceedings of the International Conference on Agile Software Development and its accompanying workshops and (ii) backward search in three secondary studies [3, 7, 5]. The search was performed using dblp, a computer science bibliography. Within the manual search, only publications from the International Agile Software Development Conference were looked at. Other relevant sources were already covered through the automated search with the search engines described in Table 4.1. The secondary papers by Uludat et al. [7] and Dikert et al. [3]

were chosen since they both provided a good overview of publications in large-scale agile software development. Therefore it was ensured that no publications in this field are missing. To ensure completeness in the area of metrics in agile software development, a backward search in the SLR of Kupiainen et al. [5] was performed. The automated search was then performed to assess the quality of the search strings and improve their effectiveness. This search was performed in SCOPUS, and the results were compared with the papers identified by Kupiainen et al. [5], Uludat et al. [7] and Dikert et al. [3] to ensure completeness.

Identification of venues or databases

One important step in performing a literature review is identifying related venues and databases that should be searched. To maximize the number of identified sources, databases that are often used in scientific writing in agile and large-scale agile software development have been chosen. The six databases chosen are listed in 4.1. Furthermore, it is necessary to decide which parts of an article should be searched [16]. In the case of this master thesis, the data to be searched include title, abstract, and keywords.

ID	Search engine	Link
DB1	Scopus	https://www.scopus.com
DB2	IEEE Xplore Digital Library	https://ieeexplore.ieee.org/Xplore/home.jsp
DB3	ACM Digital Library	https://dl.acm.org/
DB4	Science Direct	https://www.sciencedirect.com
DB5	Web of Science	https://www.webofknowledge.com
DB6	Association for Information Systems (AIS) eLibrary	https://aisel.aisnet.org/

Table 4.1.: Search engines used

Identification of study types and search strings

To remove irrelevant or low-quality studies, subject and article type should be included in the search strings [16]. The definition of the search string was performed in iterations. Firstly different categories were defined (e.g., agile software development, metrics, goals). For each of those categories, synonyms were identified. The categories and synonyms were then combined with logical operators. In compliance with the respective syntax, the resulting strings were then put into the databases (cf. Table 4.1).

Time span included in the search

The main automated search was carried out on December 21st, 2020, after the preliminary search's refinement process had been successfully completed. This date marked the upper boundary of the period that was searched. The start of the search period was February 2001, the creation date of the Agile Manifesto.

4.1.2. Inclusion and exclusion criteria

In order to decide which publications to include in the literature review, inclusion and exclusion criteria were defined. These are presented in Table 4.2. Criteria C1, C3, and C8 were inspired by the systematic literature review of Kupiainen et al. [5] as this is a very relevant related work on metrics in agile context (cf. Chapter 3).

Metrics in agile software development

In this SLR, only papers that described metrics or goals in the context of agile software development were included. Following the example of Dikert et al. [3], publications with the topic of agile manufacturing were excluded.

Industrial context

All studies identified in the filtering process have been mapped to one of the three types:

1. metrics used in industry,
2. metrics proposed by research and validated in industry,
3. and metrics proposed by research.

Within this SLR, only publications regarding metrics or goals in an industrial context have been included. This was decided after the data extraction phase of this study due to the high number of identified sources concerning agile software development metrics and will further be explained in Section 4.1.7.

Metric information amount

Within this SLR, only publications that provide data about goals or metric usage or additional information regarding metrics are included. This could be the description of challenges, success factors, reasons for metric use, stakeholders, and advantages and disadvantages of metrics or goals.

Language

Only publications written in the English language have been included in this SLR.

Publication Date

Only papers that have been published after the publication of the Agile Manifesto in February 2001 have been included. The upper boundary for the publications is January 2021.

Publication Type

To obtain publications with scientific rigor and relevance to agile and large-scale agile software development, only conference publications, journal publications, early access articles, and experience reports were included. As not all databases described in 4.1 offered the option to include filtering of research types in the search string, the filtering process has been performed afterward. Within this filtering process, review articles, short communications, conference reviews, book chapters or reviews, magazines, and newsletters have been excluded.

Access

Only publications that could be accessed in full text with the access rights by the Technical University of Munich have been regarded for this SLR.

Quality

In order to ensure the overall quality of research, only papers that did not have severe issues with grammar and vocabulary have been included.

4.1.3. Definition of search terms

The definition of search terms was performed in a structured manner. First, the topic was divided into four groups: (i) agile software development, (ii) large-scale agile software development, (iii) metrics, (iv) and goals. These were then connected with logical operators. The resulting structure was:

> *(((Agile software development AND Large-scale development) OR Scaling agile frameworks) OR Agile software development) AND (Metrics OR Goals)*

The design of the search terms regarding agile and large-scale agile software development was inspired by the systematic mapping study of Uludag et al. [7], who defined search terms for the categories of agile software development, large-scale development, and scaling agile frameworks. The set of agile software development search terms is connected with the large-scale development set with an AND operator in the search string. The set of identified scaling agile frameworks by Uludag et al. [58] (see Table 4.3) are then connected to the previous sets with an OR operator. This then results in a set of large-scale agile software development publications. As this SLR is also interested in metrics in agile software development, the set of agile software development is connected with an OR operator to the previous sets as well. To only retrieve publications concerning information about metrics or goals, these sets have been combined with an OR connector and finally added with an AND operator to the rest of the sets. For identifying search terms about metrics, the SLRs of Kupainen et al. [5] and the other related work described in Chapter 3 were used [4] [6] [56] [42]. In order to identify goals in large-scale agile software development, synonyms of the word "goal" were identified, resulting in the set described in 4.4. The keywords for all sets were put in the title, abstract, and keywords field in the search engines. The only exception being goals, which

ID	Criteria	Inclusion criteria	Exclusion criteria
C1	Metrics in agile software development	Papers that present metrics and goals in an agile software development context are included.	Studies that are related to agile manufacturing are excluded.
C2	Industrial context	Publications that present the use of metrics in an industrial context are included.	Publications, where metrics are proposed by research and are only validated in an industrial context, are excluded.
C3	Metric information amount	Papers that provide data about metric usage and additional information regarding metrics are included.	Papers that only mention metrics but do not provide any additional information about their usage, challenges, or success factors are excluded.
C4	Language	Papers, written in the English language, are included.	Articles that are not written in the English language are excluded.
C5	Publication type	Experience reports, conference publications, journal publications, and early access articles are included.	Papers that are published in the form of abstracts, book chapters, book and conference reviews, grey literature, magazines, newsletter, short communications, talks, technical reports, and tutorials are excluded.
C6	Publication date	Papers, published between February 2001 and 20th of January 2021, are included.	Papers that have been published before February 2001 or after January 2021 are excluded.
C7	Access	Publications that could be accessed in full text with the rights provided by the Technical University of Munich are included.	Publications that could not be accessed in full text by access rights provided by the Technical University of Munich are excluded.
C8	Quality	Publications with correct grammar and vocabulary are included.	Publications that have issues with grammar and vocabulary, that are therefore hard to understand are excluded.

Table 4.2.: Inclusion and Exclusion criteria

were only searched for in the title or keywords part of the publications. This was done to reduce the number of identified papers in the search, as abstracts include a description of the research objectives, and therefore a lot of false results occurred. The main search was

performed with the databases described in Table 4.1. As they all offered different syntax, the search strings for DB1, DB2, DB3, DB5, and DB6 have been defined with the keywords presented in Table 4.4. Those are shown in Appendix A.1. The search string for DB4, Science Direct, is not mentioned here, as it only offers a graphical interface that only allows for eight Boolean operators to be entered.

4.1.4. Overview of the preliminary and main search

In this section, the filtering process that was performed in the preliminary and main search is described. The filtering process describes the approach from executing the search strings in the database to identify relevant publications. Figure 4.1 gives an overview of the number of papers identified in both, main and preliminary search and the resulting set of publications after removing duplicates. The filtering process of the resulting set is visualized in Figure 4.2 and includes 110 primary studies.

4.1.5. Execution of the preliminary search

Within the preliminary search, two steps were performed: a manual and an automated search. In the manual part of the search, 14 publications from the International Agile Software Development Conference were identified as relevant by title. Furthermore, 65 papers were identified by backward search in the studies by Uludag et al. [7], and Kupiainen et al. [5]. With the automated trial search conducted in the selected databases, 125 publications were identified in total.

4.1.6. Execution of the main search

The execution of the main search identified 17042 publications in total. The main reason for this large number of results lies in the usage of Scopus. Within this database, 8349 publications were identified. A further reduction of the results by filtering within the database was not possible, as every filtering possibility led to the loss of relevant studies. Therefore, the filtering process started with 17042 studies, which were reduced to 13455 after removing duplicates. They were then merged with the set of the preliminary search, resulting in a set of 1350 publications.

4.1.7. Filtering process performed

The filtering process was performed in six levels: duplicates, meta-data, abstract, full-text access, full-text, and final decision as described in Table 4.5. The duplicate removal is visualized in Figure 4.1, and the filtering process of the resulting publications is presented in Figure 4.2. As a first step, duplicates found between databases and main and preliminary search were excluded, resulting in 13462 papers. Then publications were classified as relevant and irrelevant according to meta-data. Two researchers performed this step to ensure that no relevant publications were excluded. In this step, 108 conflicts between researchers occurred, which could be resolved, ending in 3162 sources. Due to the large number of publications

Framework	Methodologist	Publication date
Crystal Family	Alistair Cockburn	1992
Dynamic Systems Development Method	Arie van Bennekum	1994
Agile Project Framework for Scrum		
Scrum-of-Scrums	Jeff Sutherland and Ken Schwaber	2001
Enterprise Scrum	Mike Beedle	2002
Agile Software Solution Framework	Asif Qumer and Brian Henderson-Sellers	2007
Large Scale Scrum	Craig Larman and Bas Vodde	2008
Scaled Agile Framework	Dean Leffingwell	2011
Disciplined Agile 2.0	Scrott Ambler	2012
Spotify Model	Henrik Kniberg, Anders Ivarsson, and Joakim Sundén	2012
Mega Framework	Rafael Maranzato, Marden Neubert, and Paula Heculano	2012
Enterprise Agile Delivery and Agile Governance Practice	Erik Marks	2012
Recipes for Agile Governance in the Enterprise	Kevin Thompson	2013
Continuous Agile Framework	Andy Singleton	2014
Scrum at Scale	Jeff Sutherland and Alex Brown	2014
Enterprise Transition Framework		2014
ScALeD Agile Lean Development	Peter Beck, Markus Gärtner, Christoph Mathis, Stefan Roock and Andreas Schliep	2014
eXponential Simple Continuous Autonomous Learning Ecosystem	Peter Merel	2014
Lean Enterprise Agile Framework		2015
Nexus	Ken Schwaber	2015
FAST Agile	Ron Quartel	2015

Table 4.3.: Large-scale agile frameworks identified by Uludag et al. [58]

Set	Search term
Large-scale development	large-scale OR scaling
Agile software development	(agile OR agility OR extreme programming OR XP OR feature driven development OR FDD OR scrum OR crystal OR pair programming OR test-driven development OR TDD OR leanness OR lean software development OR lean development OR LSD) AND NOT manufacturing
Scaling agile frameworks	Crystal Family OR Dynamic Systems Development Method Agile Project Framework for Scrum OR Scrum-of-Scrums OR Enterprise Scrum OR Agile Software Solution Framework OR Large Scale Scrum OR Scaled Agile Framework OR Disciplined Agile 2.0 OR Spotify Model OR Mega Framework OR Enterprise Agile OR Delivery and Agile Governance Practice OR Recipes for Agile Governance in the Enterprise OR Continuous Agile Framework OR Scrum at Scale OR Enterprise OR Transition Framework OR ScALeD Agile Lean Development OR eXponential Simple Continuous Autonomous Learning Ecosystem OR Lean Enterprise Agile OR Framework OR Nexus OR FAST Agile
Metric	measur* OR metric OR diagnostic OR monitor* OR quantitative model OR indicator OR performance indicator OR KPI OR reporting
Goal	aim OR target OR object* OR goal

Table 4.4.: Keywords used in automated search

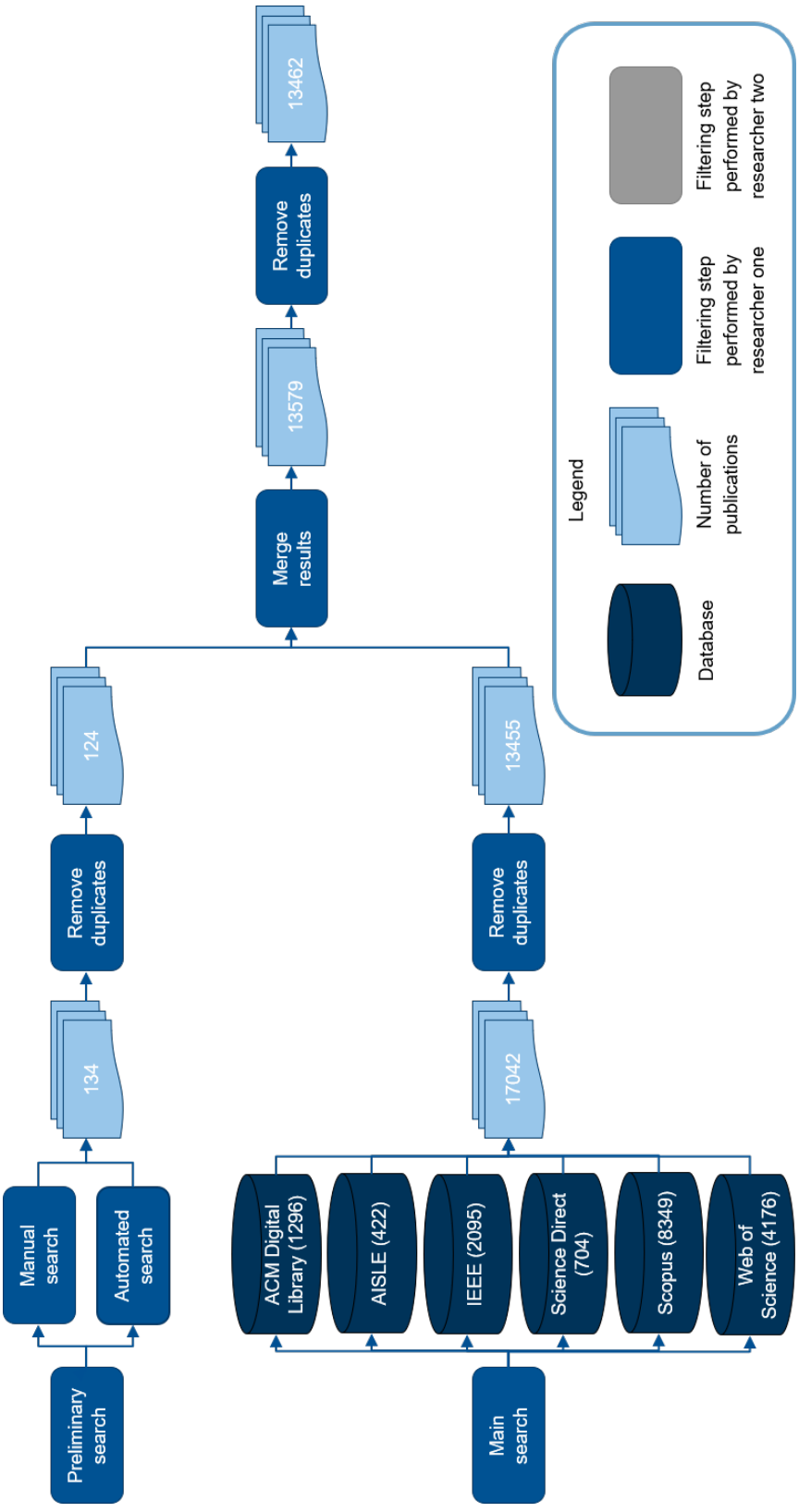

Figure 4.1.: Number of publications identified in the primary and main search

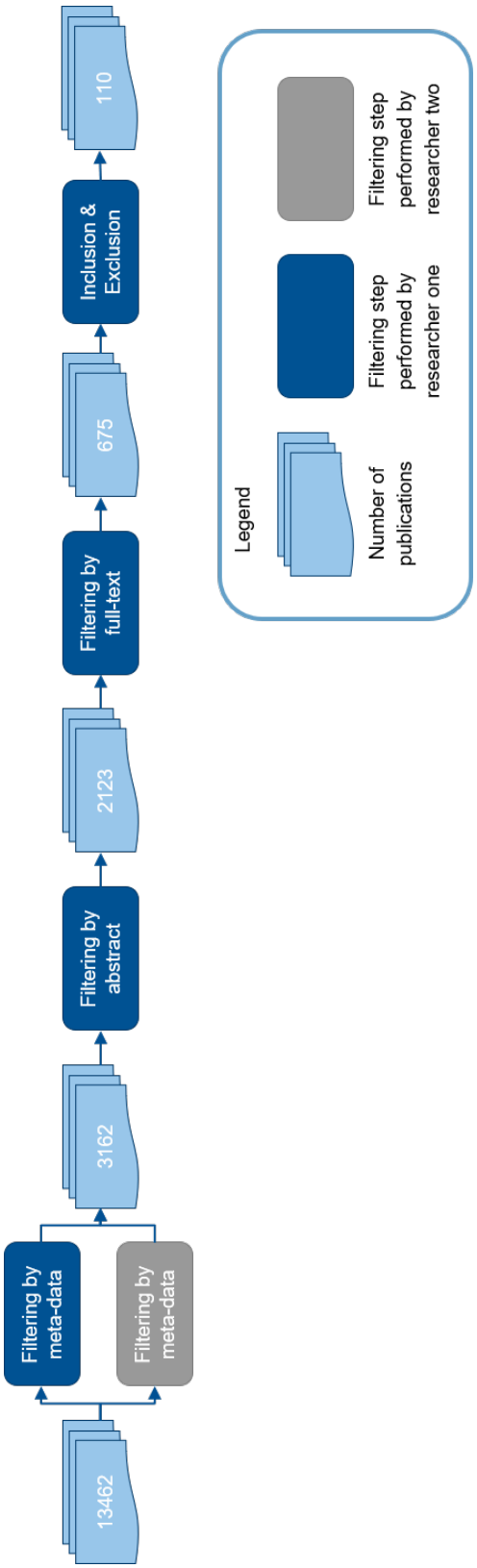

Figure 4.2.: Number of publications per filtering step

filtering by abstract, full-text access and full-text were only performed by researcher one. To support researcher one and ensure good quality decisions in the filtering process, random samples of 50 publications were mapped by researcher two in regular intervals. Conflicts in decisions about relevancy were then discussed and resolved, resulting in the final identification of 675 publications. These publications were classified as relevant due to the Inclusion and Exclusion criteria C1, C4 - C8 described in Table 4.2. The data extraction process, as described in Section 4.2 was then performed on this set of sources. Due to the high number of identified papers, the inclusion and exclusion criteria C2 and C3 were then defined, resulting in a final set of 110 publications, listed in Table 4.1.

ID	Filtering step	Description
L1	Duplicate	Duplicates occurred between databases and between main and preliminary search were removed. This was performed by hand in an Excel sheet, where all titles were sorted by alphabet and duplicates were marked by hand.
L2	Meta-data (title, keywords, publication type)	Identify relevance according to the title, keywords, and publication type
L3	Abstract	Identify relevance according to abstract
L4	Full-text access	Only papers, that could be accessed in full-text were considered relevant
L5	Full-text	Identify relevance according to the full text
L6	Final decision	Identify relevance according to data extraction performed

Table 4.5.: Overview of filtering steps

4.2. Data extraction

After the identification of studies with the main and preliminary search, the data extraction process was conducted. Within the data extraction process, sixteen criteria have been created, described in Table 4.6. The data extraction process was performed iteratively while excluding publications step by step. The exact procedure is described in the following. Part of the data extraction was performed on the set of 675 publications identified after full-text filtering. This included the categories C1-C6. Due to the large number of publications found during this literature review, only publications that describe the usage of metrics in the industry are considered relevant. The 200 resulting studies were then used for data extraction of all criteria described in Table 4.6. In the last iteration of the data extraction process, only papers with a quality mapping of 2 or 3 in the goal or metric category (described in the subsection C5: Quality of study regarding metrics and C6: Quality of study regarding goals) were included. This resulted in 110 primary studies, with which another data extraction round was

performed to classify the categories further and enhance the quality of the data extraction. In the following, a description of each category is provided.

C1: Year

The year was extracted from the BibTex citation retrieved from google scholar.

C2 and C3: Publication channel and publication type

The publication channel of the studies has been extracted from the BibTex citation. This information could be directly extracted, and only slight modifications were needed to categorize the publication channels. For conferences, the year of the conference as well as the enumeration of the conference was removed. The identified publication channels were analyzed and grouped into the three categories journal, conference, and workshop to retrieve the publication types.

C4: Research approach

In order to identify the research approaches used, the classification scheme of Rodríguez et al. [59] was used. The definition of the approaches action research, case study, industry report (here referred to as *experience report*), mixed methods, and survey are presented in 4.7.

C5: Quality of study regarding metrics

This classification scheme was adapted from the systematic literature review of Kupiainen et al. [5]. The publications were classified into quality levels ranging from -1 to 3, where -1 means no information about metrics and three refers to a good amount of relevant information about metrics and metric use. The exact definition is described in Table 4.8.

C6: Quality of study regarding goals

Similar to the criteria C6, the quality of study regarding goals was classified as shown in Table 4.9. For the definition of this classification scheme, the quality classification of Kupiainen et al. [5] was adapted to the context of goals.

C7: Organizational context

For each publication, the organizational context studied has been extracted by full-text filtering. This includes the agile method, business area, organization size, year the agile transformation has started, number of teams, and team size.

ID	Category	Description
C1	Year	The year the publication was published. This was extracted from the BibTex citation in Google Scholar.
C2	Publication channel	The publication channel, where the study was published. This was extracted from the BibTex citation in Google Scholar.
C3	Publication type	The publication type was classified by mapping the publication channels into categories.
C4	Research Approach	The research approach used in the papers was identified by keywords, abstract and full-text. An already existing classification scheme was used for this.
C5	Quality of study regarding metrics	The quality of the study regarding metrics was mapped by reading the full text. An already existing classification scheme was used for this.
C6	Quality of study regarding goals	The quality of the study regarding goals was mapped by reading the full text. An already existing classification scheme was adapted for this.
C7	Organizational context	The organizational context from each publication was extracted. This includes the time of introduction of agile methods, agile framework used, organization size, company domain, number of teams, and team size.
C8	Agile or large-scale agile software development	For each publication, a mapping of whether this publication describes agile or large-scale agile software development was performed by reading through the full text.
C9	Metric	All metrics mentioned in the publications were extracted. Classification of those metrics took place.
C10	Metric classification	Each metric that was identified was mapped into the metric classification according to Fenton and Pfleeger.
C11	Organizational level measured	Each metric was mapped to the organizational level where it was measured.
C12	Challenges in using metrics	Challenges of using metrics were extracted from full text both for certain metrics and in general. The classification was performed.
C13	Success factors in using metrics	Success factors of using metrics were extracted from full text both for certain metrics and in general. The classification was performed.

Table 4.6.: Overview of data extraction categories

Research approach	Description
Action research	When the study explicitly states this research approach.
Case study	If the study uses a case study to answer the research questions or a theoretical concept is evaluated by application in a case study.
Experience report	The study reports industrial experiences, mostly without mentioning research questions or a research method.
Mixed methods	If the study uses multiple data collection methods.
Survey	The researchers conduct interviews or questionnaires to collect data. In a survey study a representative population is used to generalize the results.

Table 4.7.: Overview of research approaches as defined by Rodríguez et al. [59]

Quality level	Description
-1	The study does not contain any information regarding metrics and should already be excluded.
0	The study mentions that metrics are used but not which metrics
1	Only descriptions of metrics with no additional info are provided.
2	The study includes some useful information related to metrics.
3	The publication provides a good amount of relevant information regarding metrics and metric use.

Table 4.8.: Overview of the metrics quality mapping inspired by Kupiainen et al. [5]

Quality level	Description
0	The study does not contain any information about goals.
1	Only a description of goals is provided, but no reason why they can not be measured.
2	The study describes goals and also provides information about why they are not measured with metrics

Table 4.9.: Overview of the goal quality mapping inspired by Kupiainen et al. [5]

C8: Agile or large-scale agile software development

In order to identify differences in metrics and goals in agile or large-scale agile software development, each study was classified as either agile or large-scale agile. As described in Chapter 2, the large-scale agile definition of Uludag et al. [7] is used. A project is considered large-scale when there are three or more collaborating teams or 70 or more collaborating people.

C9: Metrics

All metrics mentioned in the publications were extracted. Since metrics that were calculated similarly and were used for similar reasons were named differently in the studies, a classification was performed. For example, "number of defects," "number of bugs," and "defect count" were all grouped into the category "Number of defects" to reduce complexity.

C10: Metric classification

The metrics were classified by Fenton and Pfleeger's [13] widely distributed classification framework. As described in Section 2.3 measurement is the process of assigning values to attributes or entities according to predefined rules. Entities can again be divided into three parts, which are described in the following [13]:

- Processes are software-related activities that are usually associated with a timescale.
- Products are the results of process activities, e.g., artifacts, deliverables, or documents.
- Resources are entities that are required to perform the process.

All entities mentioned above are further subdivided into internal and external attributes.

- Internal attributes can be measured by examining the entity by itself without observing its behavior. These attributes include measures like code size, complexity, or defects.
- External attributes can only be measured by looking at the entity's behavior with its environment. Regarding the above example, measures concerned with the behavior of code are, e.g., the number of defects identified by the customer. Furthermore, productivity is always considered to be an external measure of the resource people. This is because productivity is dependent on many aspects of the process and product. If productivity is measured in the form of $\frac{\text{Lines of code}}{\text{Time needed for development}}$, a process measure (effort) is combined with a product measure (size).

Further examples of this classification for each entity and attribute are displayed in the components of software measurement as defined by Fenton and Pfleeger [13] visualized in Table 4.10.

C11: Organizational level measured

Metrics were furthermore classified in a hierarchical structure. The level at which measurements are conducted was divided into five levels, which are inspired by the Scaled Agile Framework (SAFE) framework [60]. As SAFE is structured into three levels: team, program, and portfolio, the need for two more levels were identified after data extraction. So the individual level and the enterprise level were added. The exact definition of each level is described below.

- **Individual level** describes measures that are related to one individual.
- **Team level** includes all measures that are conducted on the team level.
- On the **program level** multiple teams and stakeholders work together on a solution [61].
- On the **portfolio level** decisions about multiple projects are made to optimize business value flow. It is made sure, that the enterprise meets its strategic objectives [61].
- **Enterprise level** describes measurements that take the whole organization into account.

C12: Challenges of using and implementing metrics

The derivation of challenges followed the approach defined by Dikert et al. [3] and is described in the following. Whenever a description of challenges or problems of metrics in general or of a particular metric during usage or implementation was mentioned, the information was extracted. After all challenges were extracted, all quotations of challenges were carefully read and codes for challenges were defined. After a code category was assigned to each challenge, these were analyzed and challenge categories were identified. In a final step each challenge was then assigned to a challenge category.

C13: Success factors of using and implementing metrics

Whenever a primary study mentioned best-practices, success factors or advice on how to use or implement measurements, the information was extracted. The coding and classification of success factors was performed similarly to the process described during the extraction of challenges.

ENTITIES	ATTRIBUTES	
Products	*Internal*	*External*
Specifications	size, reuse, modularity, redundancy, functionality, syntactic, correctness ,...	comprehensibility, maintainability, ...
Designs	size, reuse, modularity, coupling, cohesiveness, functionality, ...	quality, complexity, maintainability, ...
Code	size, reuse, modularity, coupling, functionality, algorithmic complexity, control-flow structuredness. ...	reliability, usability, maintainability, ...
Test data	size, coverage level, ...	quality, ...
...	...	...
Processes		
Constructing specification	time, effort, number of requirement changes, ...	quality, cost, stability, ...
Detailed design	time, effort, number of specification faults found, ...	cost, cost effectiveness, ...
Testing	time, effort, number of coding faults found, ...	cost, cost-effectiveness, stability, ...
...	...	...
Resources		
Personnel	age, price, ...	productivity, experience, intelligence, ...
Teams	size, communication level, structuredness, ...	productivity, quality, ...
Software	price, size, ...	usability, reliability, ...
Hardware	price, speed, memory size, ...	reliability, ...
Offices	size, temperature, light, ...	comfort, quality, ...
...	...	...

Table 4.10.: Components of software measurement as defined by Fenton and Pfleeger [13]

5. Results

5.1. Overview of the primary studies

This Section provides an overview of the 110 primary studies.

Study distribution

To give an overview of the popularity of the research area of metrics in large-scale agile software development, the year of the publications was collected. The number of publications identified per year is visualized in Figure 5.1. All primary sources have been published between 2001 and 2021. Overall, an increase in the number of publications can be seen in Figure 5.1. The minimum number of published studies can be found between 2001 and 2004, with only one study per year. The maximum number of publications per year was found to be in 2019, with 10 identified primary studies. The highest decrease of publications was in 2014, where only three sources were identified. Only two studies with a publication date in 2021 are included since this SLR stopped the search process after January 2021.

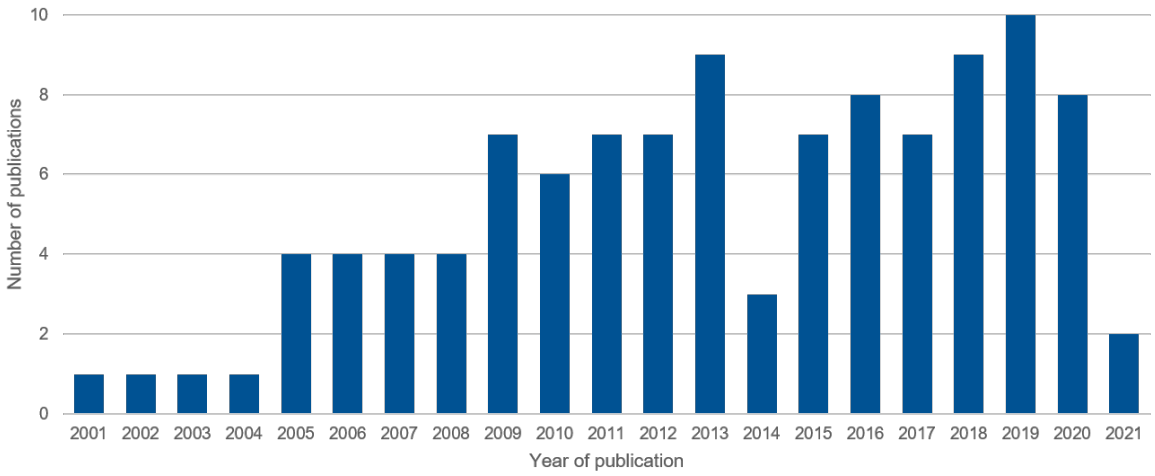

Figure 5.1.: Distribution of publications per year

Distribution of publication types

Figure 5.2 shows the distribution of publication types. Within the 110 primary sources, three distinct publication types were found. Seventy-three percent of the studies were published in conferences, 20 percent in journals, and seven percent in workshops. Figure 5.3 displays the distribution of primary sources over conferences. The 80 studies published in conferences were published in 36 different conferences. The most popular conferences were the Agile Conference (AGILE) with 17 primary sources, the Hawaii International Conference on System Sciences with 13 publications, the International Conference on Product-Focused Software Processes with five publications, and the Agile Development Conference with four studies. In the remaining conferences, less than three publications each were published. The distribution of primary sources over the fifteen journals as visualized in Figure 5.4 is spread equally. Only in the journals IEEE Software (4 studies), the Journal of Software Evolution and Process (3 studies), and the Information and Software Technology journal (3 studies) more than one primary study was identified. Figure 5.5 shows the distribution of primary studies published in workshops.

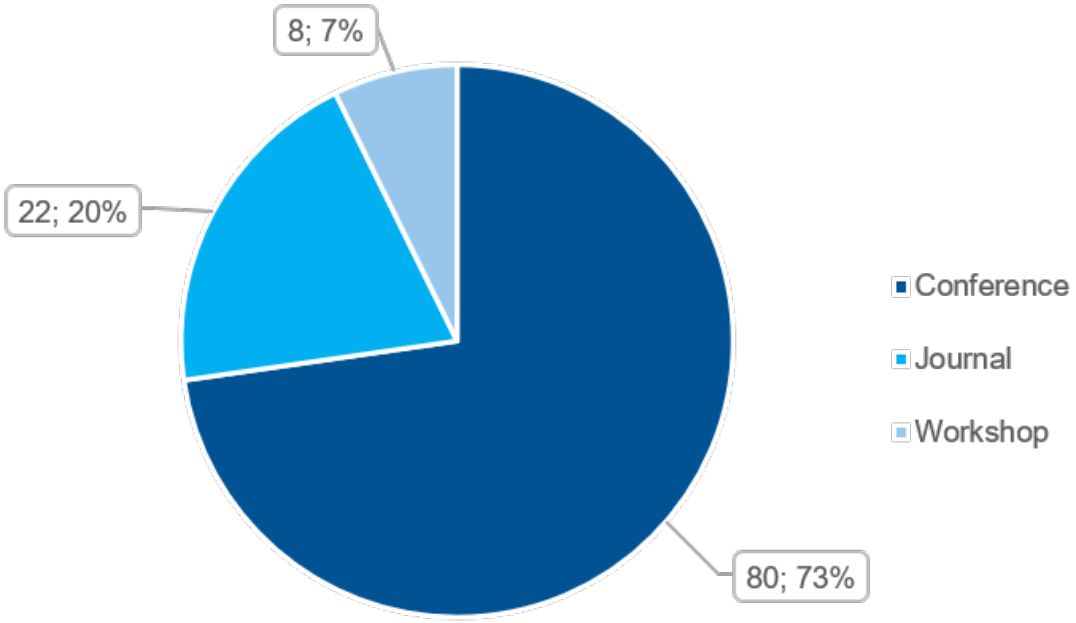

Figure 5.2.: Distribution of publication types

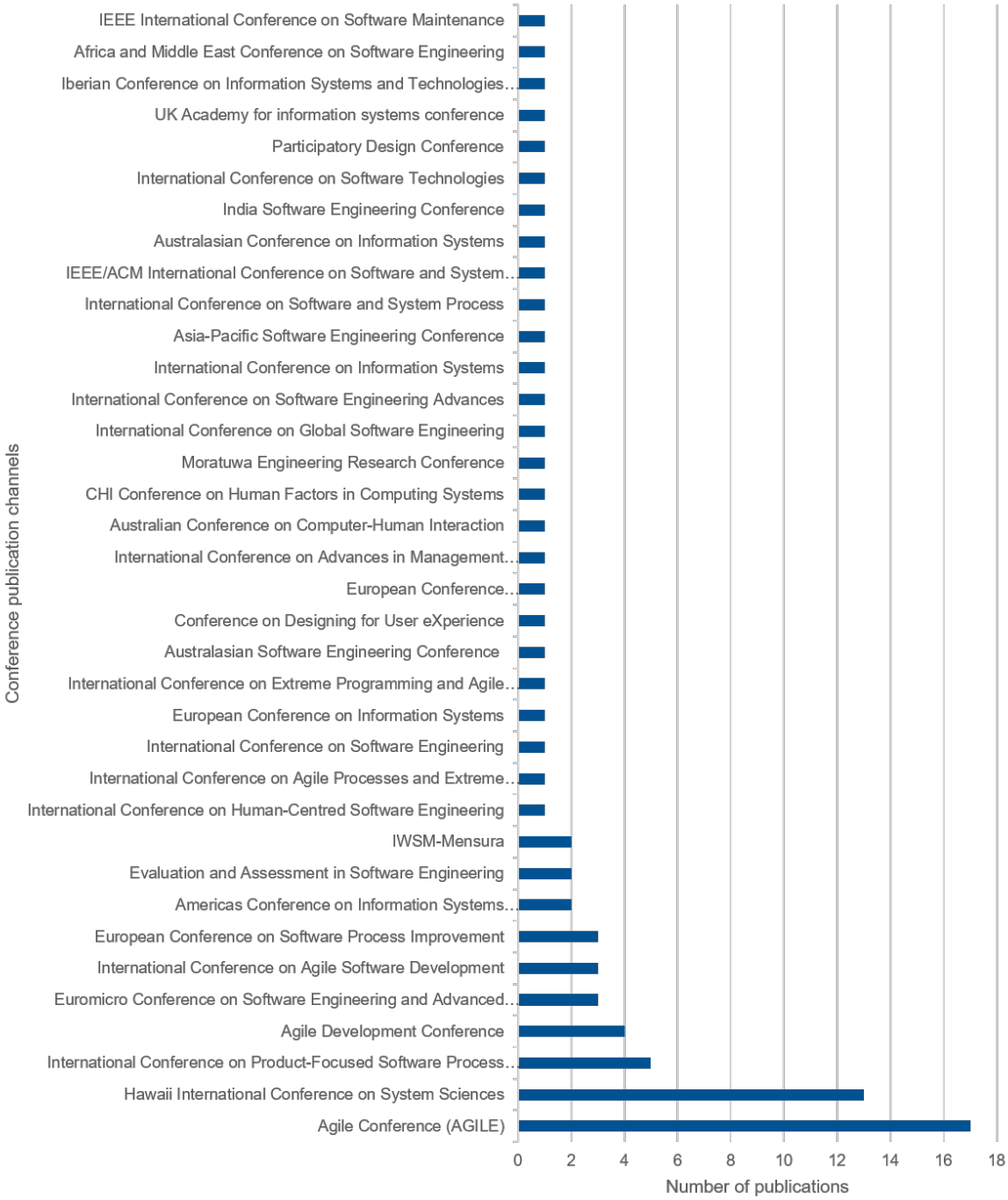

Figure 5.3.: Distribution of conference publication channels

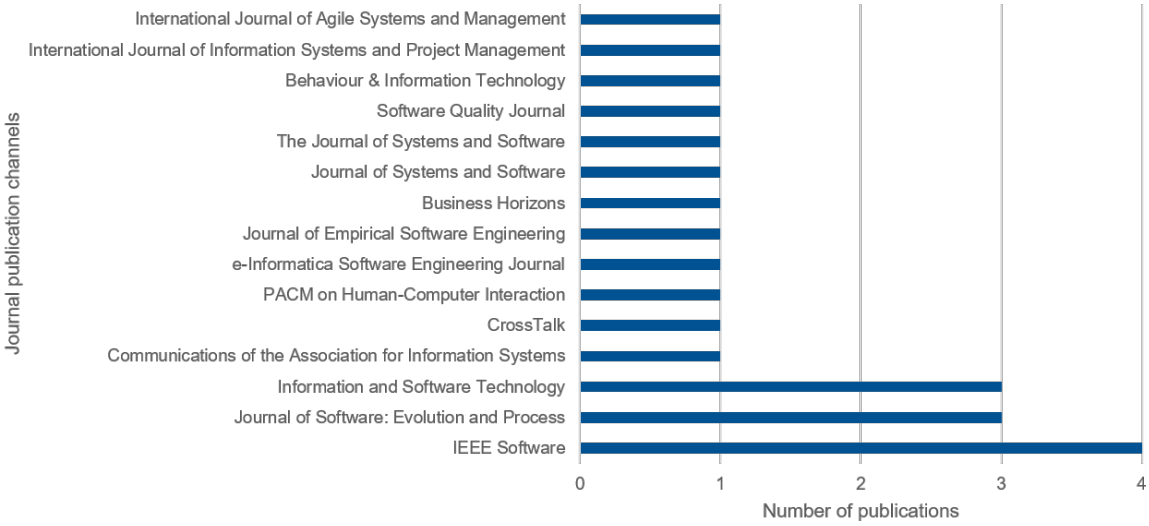

Figure 5.4.: Distribution of journal publication channels

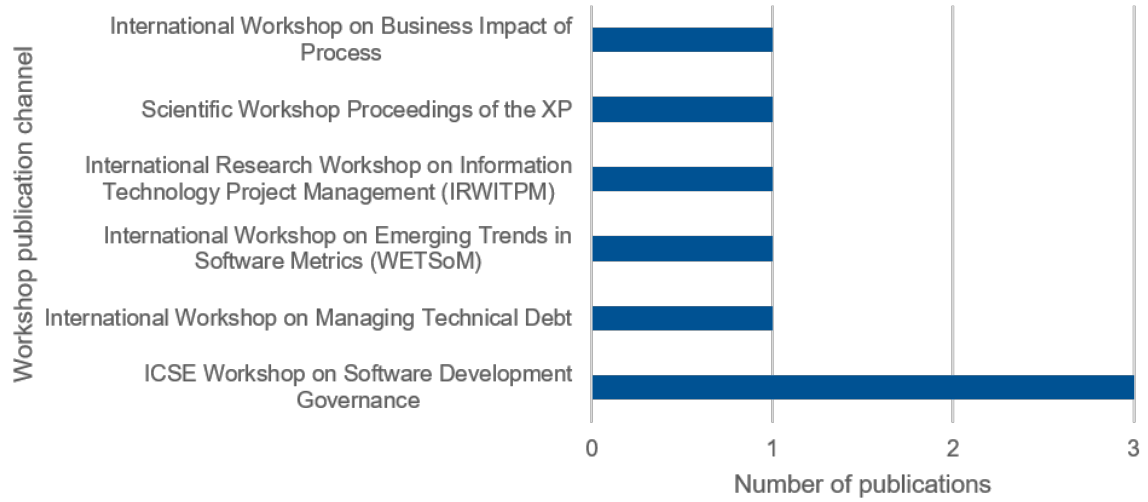

Figure 5.5.: Distribution of workshop publication channels

Research approaches

Five different research approaches were identified in the 110 primary sources. As visible in Figure 5.6, 51 percent (56 studies) of the identified studies used the case study approach. Experience reports make up the second-largest group, with 33% (36 studies). In 13 percent of publications surveys were used for investigating goals and metrics in large-scale agile software development. Mixed method design (3 studies) and action research (1 study) are clearly in the minority.

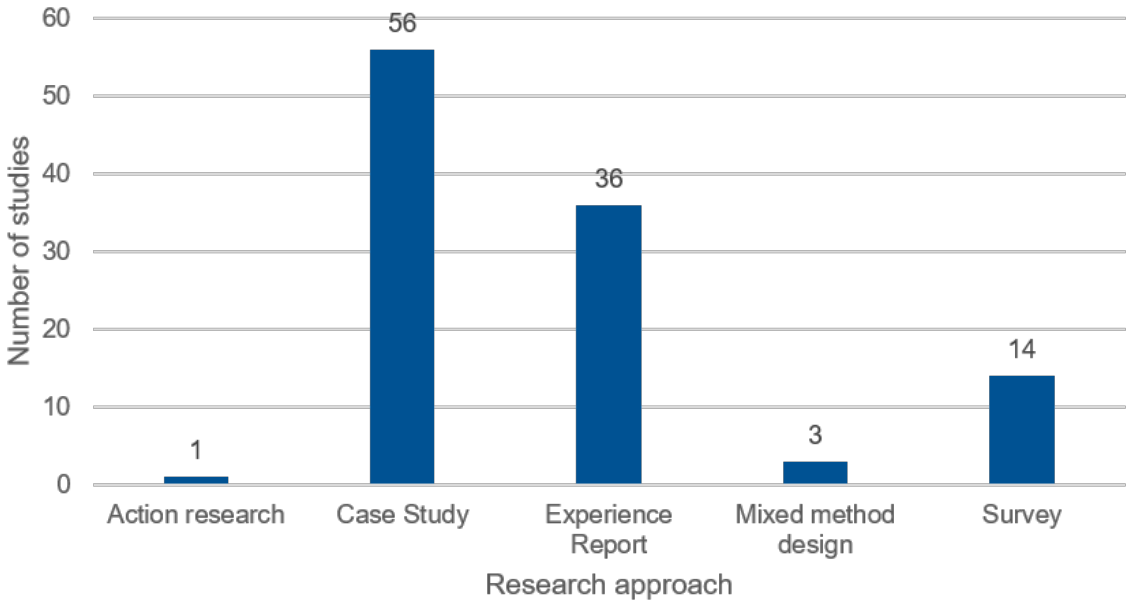

Figure 5.6.: Distribution of research approaches

Context of primary studies

Table A.2 shows the organizational context of the primary studies. Forty-five out of the identified 110 primary studies are considered to be large-scale agile software development. As some primary studies conducted multiple case studies, overall 129 industrial settings were examined. The distribution of agile frameworks mentioned in the primary studies is visualized in Figure 5.7. Scrum was the most frequently used framework (30%), followed by Extreme Programming (11%) and a mixture of multiple agile approaches (11%). Furthermore, a large number of publications (23%) did not specifically state the agile methodology that was used.

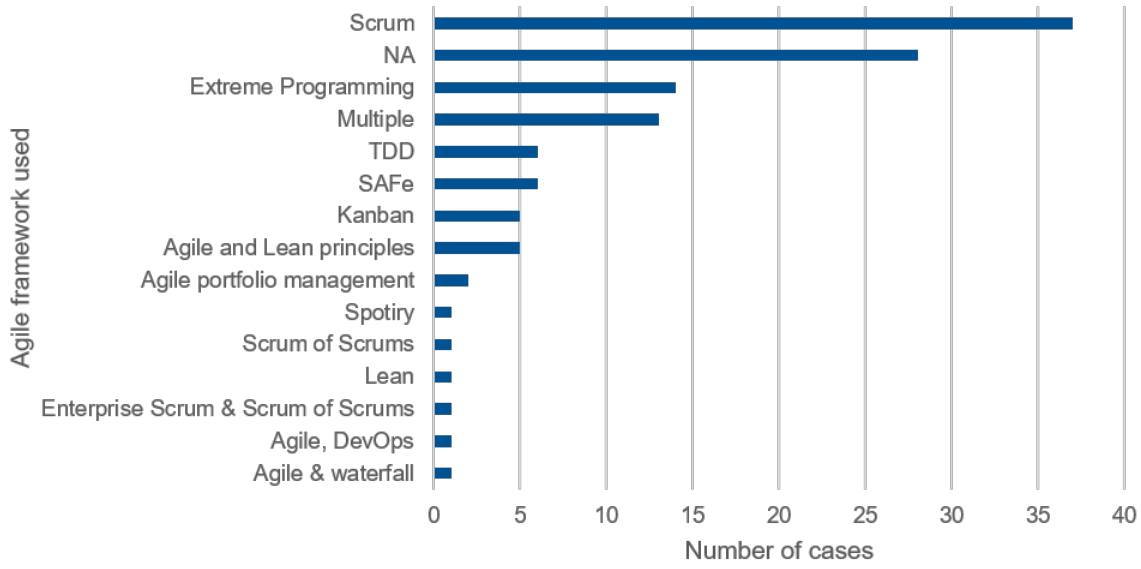

Figure 5.7.: Distribution of agile frameworks mentioned in the primary studies

5.2. Metrics and goals in large-scale agile software development

The first research question is:

What are goals and metrics in large-scale agile software development?

To answer this question, all 828 extracted metrics were classified and compared. After grouping similar measures that are named differently, 273 different measures were identified. The full list of metrics, their frequency of occurrence, in which study they were mentioned, and their definition can be found in Appendix A.4. In order to provide a better overview of this large number of metrics, two further classifications have been performed. First, the organizational level at which measurements were conducted was identified. Afterward, a classification of metrics according to Fenton and Pfleeger [13] was performed, to provide an overview about which entities and attributes are measured. Lastly, the goals mentioned in large-scale agile software development are describes.

5.2.1. Organizational level measured

Figure 5.8 displays the distribution of metrics over the levels at which measurements were conducted. Most measurements were conducted at the team level (69%, 571 measurements). Out of the 571 mentions of measurements, 209 different metrics were used. Program level was the second most measured level, with 26% (215 measurements) across 121 different metrics. Only three percent of measures (28 measurements of 20 metrics) took place on portfolio level. On the individual level, only six different metrics were used, and eight measurements were conducted (1 percent). The least amount of measurements (1%, 6 measurements across 6

metrics) took place on the enterprise level. The twenty most used metrics in each individual level are displayed in Table 5.1.

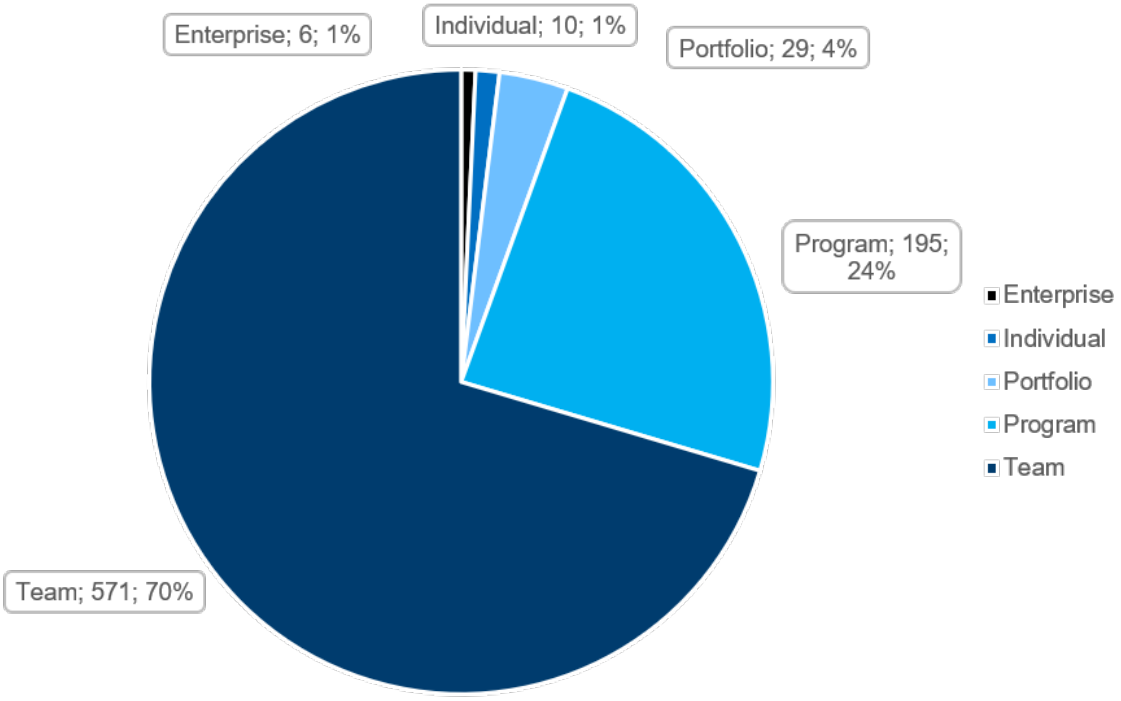

Figure 5.8.: Distribution of measures across organizational levels

Enterprise level

On the enterprise level, only six different metrics were used. Out of those six metrics, five were mentioned in two experience reports from Citrix [S1] [S2]. The company used Enterprise Scrum and Scrum of Scrums as a large-scale method and had 28 teams working across six products. They use enterprise story points to estimate the effort for implementing Enterprise Backlog Item (EBI)s. Those are first estimated by product board members, then by team leads, and finally by teams. After the estimation by the product board members is performed, the EBIs are ordered by priority. The estimated effort is hidden to keep the estimation of the team leads unbiased. After the team lead has estimated the EBI, the estimations are compared. In case of differing estimations, the product board members need to re-estimate. The same approach is then followed for the team estimations. Furthermore, Citrix measures velocity on the enterprise level as story points implemented per month by the whole engineering department. They also introduced operations velocity to resolve the bottleneck of projects being stuck in deployment services due to not enough resources available in operations. The last metric used on enterprise level by Citrix was company productivity, measured as Net

Present Value (NPV) of all projects divided by effort spend on all projects. [S1] The success of reducing time-to-market on the enterprise level was measured by calculating the average release duration of all projects [S2]. Petrobras, an oil and gas company, used a build status monitor to get provide an overview of the build state of each project and team in the company [S3].

Measurement level	Twenty most used metrics
Enterprise	Enterprise Story Points (1), Velocity (1), Operations velocity (1), cycle time (1), build status monitor (1), efficiency regarding cost (1)
Portfolio	Return on investment (ROI) (4), Net Promoter Score (NPS) (3), application downtime (2), customer satisfaction (2), customer Experience Index (1), velocity (1), stakeholder happiness (1), business value delivered (1), investment allocation report (1), results coming period (1), portfolio wall (1), cycle time (1), results past period (1), defect count (1), Net Present Value (NPV) (1), dependency visualization (1), structural quality (1), estimation accuracy (1), # of active users (1), generated income (1)
Program	Defect count (15), # of tests (8), code coverage (8), estimation accuracy (7), lines of code (LOC) (5) # of closed defects (5), technical Debt (5), lead time (4), velocity (4), # of completed tasks (4), churn (4), throughput (3), task board (3), # of passed tests (3), test review yields (3), integration status (2), dependency visualization (2), sprint burndown chart (2), # of tests planned (2), # of defects found by customer
Team	Velocity (41), defect count (38), estimation accuracy (21), complexity (21), code coverage (19), time to fix defects (18), burndown chart (16), lines of code (LOC) (12), deferred defects (11), # of tests (11), cycle time (11), defect density (11), Net Promoter Score (NPS) (8), task distribution 8, # of completed tasks (8), # of closed defects (7), task completion time (6), team satisfaction (6), availiable capacity (6), cost (6)
Individual	Available capacity (3), individual effort remaining (1), individual effectiveness (1), developer reputation (1), time to fix defects (1), employee cost (1)

Table 5.1.: Distribution of metrics across organizational levels

Portfolio level

Within the primary studies, 21 metrics were measured at the portfolio level. The most mentioned measurements were Return on Investment (ROI) [S4] [S5] [S6] [S7] and Net Promoter Score (NPS) [S8] [S6] [S9]. Application downtime [S9] [S10], and customer satisfaction [S11] [S9] were mentioned in two studies. Eight of the 21 metrics on portfolio level were mentioned in the Case Study of Kopripaara et al. [S9] at a large banking organization. They mea-

sured customer satisfaction by using the metrics customer satisfaction survey, the Customer Experience Index, the net change in active users, and the NPS. The financial value of the portfolio was measured with generated income, sales per service channel, and the amount of turnover. Application downtime per project and the number of reported incidents per project were calculated to measure IT stability. [S9] Stettina and Schoemaker [S12] described six metrics conducted at portfolio level: velocity, stakeholder happiness, results coming and past period, portfolio wall, and dependency visualization. Business value delivered was used at two financial companies [S13]. Integraph, a software developer, validated their investment decisions by calculating a Project allocation and an Investment allocation report, where they compared planned and actual allocation of effort per product [S14]. Fannie May consulting used a structural quality heat map that visualized both quality and size of products over the portfolio, sub-portfolio, and product level [S15]. Lastly, cycle time and customer satisfaction were measured at Capital One Auto Finance [S11].

Program level

On the program level, the second most measurements were conducted. Since 121 different metrics were used on this level, only the 20 most mentioned metrics will be discussed in this section. The complete list of all metrics found per level can be found in Appendix A.3. With 15 citations, the number of defects was by far the most measured metric at the program level. Code coverage and number of tests are the second frequent measurements with eight nominations. For their project with 150 developers, Hewlett Pacard used multiple measures of the number of defects, code coverage, churn as the number of LOC or files added, the number of passed tests, and a task board [S16].
StarSoft [S17], Petrobras [S3], Ericsson AB [S18], and Hewlett Pacard [S16] use the size measure Lines of Code (LOC). Four companies measured the efficiency of defect removal across teams in the number of closed defects [S16] [S19] [S20] [S21]. Burndown charts were frequently used (four mentions) to display the burndown of product backlog items within a project across teams [S12] [S22] [S17] [S1]. Technical debt was measured on the program level by three companies [S19] [S3] [S23]. Siemens [S23], and Petrobras [S3] both describe multiple measures related to technical debt. These include the total amount of technical debt and the technical debt board, which visualizes the importance of the removal of technical debt. Lead time, the number of completed tasks, churn, and velocity were each measured four times on the program level. In a case study, Gustavson [S24] describes the use of program boards in two companies that are used to visualize dependencies between teams and tasks.

Team level

With 209 different metrics and 571 measurements conducted team level metrics were reported most frequently. On the team level, velocity was the most cited metric (41 occurrences). Velocity describes the number of completed features or stories per iteration [S25] [S26] [S27]. The number of defects was the second most frequent metric (38 citations). The third most frequently used metric was estimation accuracy, measured as the difference between estimated

effort and actual effort spend [S28] or the actual effort minus the planned effort divided by the estimated effort [S29]. The complexity of code was measured in terms of the number of classes [S30] [S31] [S25], interfaces [S25], and methods [S32] [S31]. In 19 cases, companies stated the use of code coverage, which describes the amount of code covered by test cases [S31]. IONA Technology [S33] and 18 other companies measured the time needed to fix defects. Further metrics regarding defects on the team level were the number of deferred defects (11 occurrences), defect density (11 occurrences), and the number of closed defects (7 occurrences). Burndown charts, which visualize the amount of work scheduled in the current sprint in relation to the work already completed by the team [S34] [S35], were mentioned 16 times. Atlassian [S36] and seven other companies conduct customer surveys, with which they calculate the NPS. During the NPS customers are asked how likely they are to recommend the product to others. All user ratings on the scale from 1-10 are then grouped into one of three categories: promoters (rating of 9-10), passive (rating of 7-8), or detractors (rating of 7 or less). To calculate the NPS, the overall percentage of detractors is subtracted from the overall percentage of promoters. [S37] [S36] Furthermore, task distribution (8 occurrences), the number of completed tasks (8 occurrences), and task completion time (6 occurrences), team satisfaction (6 occurrences), available capacity (6 occurrences), and cost (6 occurrences) were measured.

5.2.2. Classification of metrics according to Fenton and Pfleeger [13]

Table 5.2 shows the classification of the identified metrics according to the categorization of Fenton and Pfleeger [13]. With 69%, process was the most frequently measured entity, followed by product 20%, and resource measures 11%. Overall, 55% of metrics measured internal attributes, and 45% were external measures.

Classification of metrics

ENTITIES	ATTRIBUTES	
Products	*Internal*	*External*
Product		Thumbs-up-rule, early signal testing scorecard (ESTS), landing zone story, maintenance effort, product downtime, average usage days vs user count, consumer feedback, customer experience index, customer satisfaction, customer value, end user feedback, instrumented scorecard, net promoter score (NPS), pain index, rating and satisfaction scores, stakeholder happiness, user performance
Code	Binary file size, code security, commit size, complexity, duplicated code, LOC, LOC/file, LOC for refactoring, # of memory leaks, RUF metrics, technical dept	Maintainability, code reliability, mainline breakage, maintenance effort, TQI score
Documentation	Amount of documentation	
Features	epic status, estimation per story point, files per user story, function points, ideal days, ideal hours/story point, pomodoro, story points enterprise story points	
Specification	Requirement readiness	Requirement clarity index
Test data	Code coverage, defect size, % of automated tests, % of files without critical/blocker quality rules, # of tests planned	Test rig availability, execution time per test case, # of fast tests, % of fast test builds
Processes		
Constructing specifications	# of completed feature specifications, % of feature specifications delivered on time, sprint readiness	Velocity of elaboration features
Design	# of generated ideas by the team, time between the designers readiness of model and model release, time needed for non-implementation	# of collected ideas from key stakeholders, # of generated ideas from third party, # of granted patents, # of patents applications
Requirement engineering	Scope change report, time for unforeseen complexity, requirement coverage	Time spend for change based on client feedback
Estimation	Effort per use case point, # of estimated tasks, overall effort estimated, real invested effort in a story point	Consistency of relative estimation, correlation between story point and actual effort, effort spend, estimation accuracy, overdue percentage, % of estimations under the threshold of accuracy

Continuation of Table 5.2.: Components of software measurement as defined by Fenton and Pfleeger [13]

ENTITIES	ATTRIBUTES	
Implementation	Automation speed, churn, core component commits, effort to take technical debt to zero, fix time of failed builds, integration status, mean time to restore service, non-issue component commits, of closed defects, # of commits, # of completed tasks, # of features integrated, # of tasks added during sprint, # of tasks in progress at the end of the sprint, # of tasks planned and not integrated, % of tasks completed, sprint report, task completion time, task distribution, task queue time, task status, throughput, time between events	Amount of rework, bug correction throughput, CSR handling, lead time, LOC removed in refactoring, mean time to failure, # of blocking WIP, # of open issues whose created time is older than a treshold, resolved task throughput, system response time, time to fix defects, velocity, velocity vs unplanned effort rate,
Review	Commit review speed, # of iterations in code review, defects found in peer review	Average throughput time in a comment from reviewers, commit review performance, developer-tester communication cycle, defects found during design review
Testing	code coverage, defect classification, # of defects, # of defects during design review, defect density, defect density after internal delivery, defect density in daily build, defect distribution, effort spend for quality activities, # of critical defects, # of failed tests, # of GDI leaks, # of passed tests, # of tasks tested, # of tasks pending tests, # of tests, % of automated tests, % of errors identified after sprint review, % of fast test builds, % of passed tests, test automation ratio, testing stability of a project, total effort spent in testing	Blocked issued chart, bug priority distribution, deferred defects, defect identification performance, defect detection percentage (DDP), feedback time, mean time to failure, # of feedback provided by testers to developers, # of defects found by customer, # of released defects, # of reopened defects, open defect severity index, structural quality, test review yields, test-growth ratio
Development cycle	Backlog, cumulative flow diagram, delivery time and deadlines to achieve value, effort spend, integration speed, # of blocking WIP, # of function points/story points implemented, # of releases delivered on time, # of remaining tasks, # # of tasks reopened, % of features delivered on time, % of finished scope, % of releases delivered on time, project dashboard, status of release steps per sprint, remaining effort, task board, task distribution, task-type in a timeframe, work in progress	Burndown chart, burndown ratio, burnup chart, customer involvement, cost of unplanned changes, cycle time, definition of done, delivery on time, dependency visualisation, drag factor trend, epic burndown chart, epic progress report, flow efficiency, lead time, milestone success, non-compliance index, operations velocity, % of adopted work, % of finished scope, portfolio wall, project highlights, pulse metric, release burndown chart, sprint burndown chart, task distribution, throughput, velocity, volatility index, waste, working software delivery success rate
Build	Application downtime, build interval, build time, # of broken builds, # of builds, time needed for deployments	Build status monitor, build lead time, build throughput, code commits/broken builds, build performance, daily build pass through time
Resources		

Continuation of Table 5.2.: Components of software measurement as defined by Fenton and Pfleeger [13]

ENTITIES	ATTRIBUTES	
Personnel	Available capacity, bucket- and fix-it-list, contribution chart, control chart, effort remaining, employee experience, employee interaction, functional capacity evaluation, individual effort remaining, list of contributors, # of contributors, # of meetings, self-assessment of continuous improvement, team satisfaction, team status, turnover	Actual hours of team on ideal capacity, commitment index, developer reputation, employee cost, focus factor, individual effectiveness, team effectiveness, top contributors based on LOC, virtual planned hour cost (VPHC)
Customer	Amount of users, # of active users	
Economic	Cost, results past period	Business value delivered, cost adherence, delta cost, earned business value, efficiency regarding cost, forecast horizon, generated income, innovation rate, investment allocation report, results coming period, return on investment (ROI), sales, schedule performance index (SPI), targeted value increase, virtual actual hour cost (VAHC)
Software	Application downtime, tool availability [S38], # of changes introduced to tooling [S38], time for execution of system functions [S25]	
Hardware	Application server capacity [S25], database capacity [S25], response time for server [S38]	

Table 5.2.: Components of software measurement as defined by Fenton and Pfleeger [13]

Product measures

As described in Section 4.2, products are the artifacts that result from a process [37]. With 166 measurements, 20% of metrics could be classified to the product domain. Out of those measures, 70% represented internal, and 30% represented external attributes. Following the classification of Fenton and Pfleeger [13] and the adaption to the framework by Kupiainen et al. [5] the product entity was further divided into the six categories product, code, documentation, features, specification, and test data. Out of these categories, code was the most measured one (40%). Code is measured to 92% measured internally, with metrics like code complexity, LOC, and the number of duplicated code. Only five external code measures were identified. Measures related to the product as a whole made the second largest group in the product category (23%). In contrast to the other categories, only external measures were identified as a measure of product. The reason for this is that all metrics that are related to customer satisfaction are external measures of the whole product. The most frequently used metrics in this category are NPS (11), customer satisfaction (9), and consumer feedback (2). The third most common measurements were used to measure test data (22%). Internal measures of test data, which made up 84% of measurements, mainly included code coverage (24) and the number of planned tests (3). External measures of test data included the execution time per test case and the test rig availability (1). Features (10%), documentation (4%), and specification (1%) were the least measured product entities.

Process measures

As described in Section 4.2, processed are all software-related activities [37]. With 68% process is the most frequently measured entity. Within the process entity, in 49% of cases, internal attributes and in 51% of cases, external attributes were measured. Within the process entity nine categories were defined: testing (31%), implementation (28%), development cycle (23%), estimation (7%), build (4%), design (3%), review (2%), constructing specifications (2%), requirement specification (1%). Testing, which includes all measurements related to the number of tests and identified defects during testing, is measured internally in 77% of the cases. The most common internal test measures are the number of defects (54), the number of performed tests (19), and the defect density (12). External measures were deferred defects (12), the number of defects found by the customer (4), test review yields (4), and the number of reopened defects (4). In the implementation category, all measures related to the number of completed tasks, the amount of developed code, and the time to complete features during implementation are included. Fifty-seven percent of measures in the implementation category are performed on external attributes. The most frequent external attribute measure is velocity (43), followed by the time to fix defects (21) and task completion time (7). All external measures in this category are related to efficiency or productivity and are categorized as external because it describes how resources perform a process. Internal measures of implementation are the number of completed tasks (12), the number of closed defects (12), and churn (5). Measures relating to the whole development cycle were the third most frequent process category. It includes all measured that describe the scope of the whole development

process. Like the implementation category, external attributes (70%) were measured more frequently than internal attributes (30%). Internal measures were task distribution (5), the amount of work in progress (3), and the task board (3). Burndown chart (21), cycle time (14), and lead time (7) were the most identified external measures.

Resources measures

Resources, which describe everything that is required to perform a process [13], were the least measured category (11%). Resources were grouped into five categories: economic (46%), personnel (45%), software (3%), hardware (3%), and customer (2%). Economic resources include all measurements related to money and are mostly external (82%). Metrics measuring external attributes of economic value include ROI (10), business value delivered (7), and Cost Performance Index (2). Internal measures of economic value are cost (6), and results past period (2). Personnel is the second most frequent category. Personnel measures include the internal metrics available capacity (9), team satisfaction (6), and the number of contributors/developers (2). External measures of personnel are individual effectiveness (3), top contributors based on LOC (1), and the actual hours of the team on ideal capacity (1). For software, hardware, and customer only internal measurements have been found.

5.2.3. What are goals in large-scale agile software development

Figure 5.9 displays the distribution of goals identified in the primary sources. Overall 204 goals were extracted, which could be grouped into sixteen categories. With 28%, increasing quality was the most mentioned goal in the primary studies. Increasing quality includes the goal of reducing technical debt [S39], increasing code quality by refactoring [S40], and achieving IT stability [S9]. The second most frequently mentioned goal is operating at optimal capacity (17%). In order to reach that goal, companies tried to identify bottlenecks [S20], and increase the productivity of development [S41], [S42] [S9] [S43] [S33]. Increasing customer satisfaction was the third most frequently mentioned goal (15%). In a large banking company, customer satisfaction was perceived as the end goal [S9], and the product portfolio was aligned according to maximize it [S6]. Increasing transparency was mentioned 23 times (8%). At the Israeli Airforce, transparency was needed to identify whether other goals were met [S26]. Garzás and Paulk [S44] reported the need for transparency to make corrective actions possible by providing objective insights. Delivering the product adherent to schedule was stated as a goal 13 times (5%). As a goal that should be achieved with the introduction of agile software development [S45] [S46] [S47], reducing time to market was mentioned twelve times. Reducing cost (5%), having accurate estimation (4%), reducing risk (3%), increasing employee satisfaction (3%), ensuring compliance to agile methodologies (2%), learning (2%), collaboration (2%), motivating people (1%), and increasing usability (1%) were the least mentioned goals within the primary sources.

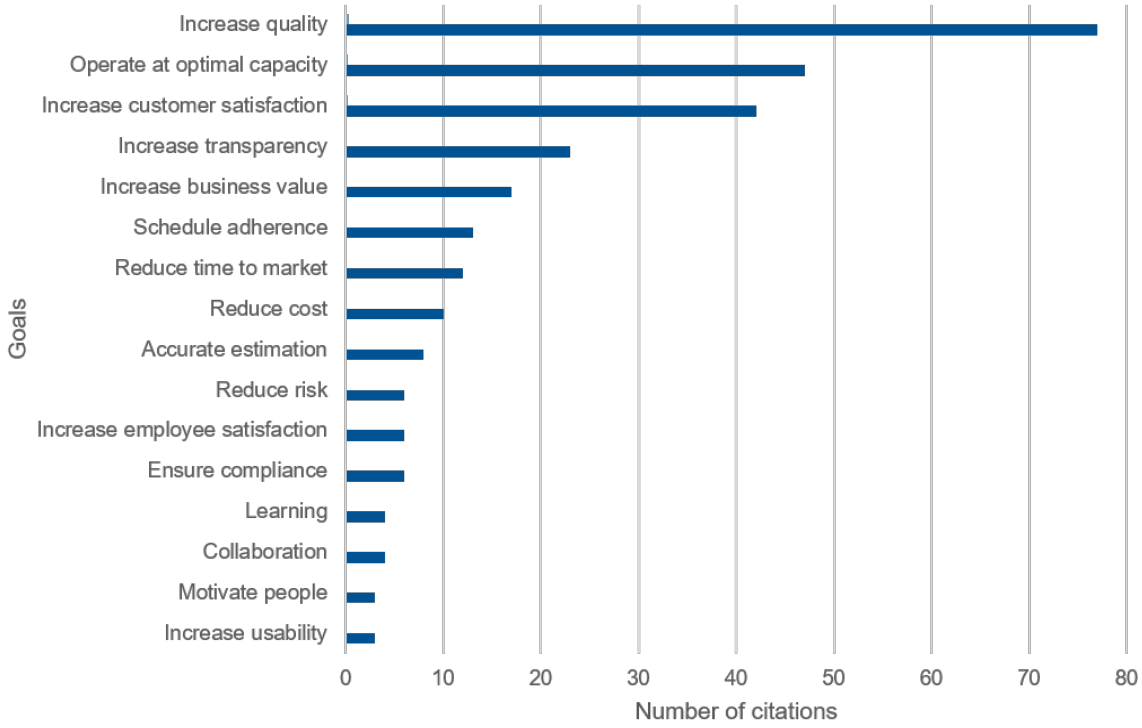

Figure 5.9.: Distribution of goals in large-scale agile software development

5.3. Challenges related to metrics in large-scale agile software development

What challenges of using and implementing metrics in large-scale agile software development have been reported?

To answer the second research question, challenges regarding metrics in large-scale agile development have been extracted from the 110 primary studies. Table 5.3 displays all of the 21 identified challenges. These were grouped into six different categories, which will be described in the following. The most often identified challenge was data collection (28%), followed by the difficulty of implementing metrics (23%), no shared understanding of metrics (18%), coordination challenges in the multi-team environment (10%), hierarchical boundaries of measurement (10%), and misinterpretation of metrics (10%).

5.3.1. Data collection

With twenty-eight percent, the difficulty of data collection was the most identified challenge within the primary sources.

Data availability

Data availability was mentioned as a challenge by seven studies. Ram et al. [S41] [S19] and Kortum et al. [S49] reported that metrics could not be developed because the necessary data was not available since no tools were used for data collection. At Adobe Systems, Green [S48] stated that metrics could not be compared because of data unavailability due to missing tools or non-transmitted data. Ram et al. [S19] and Gruschwitz and Schlosser [S25] mentioned that the company process did not produce the data needed for measurement. At a case study in a large banking company, Korpivaara et al. [S9] found out that the selection of metrics that were implemented was constrained through the limited availability of data because otherwise investment in data infrastructure would have been necessary. A lack of tools was mentioned as a challenge in three primary studies.

Time and effort needed for data collection

Seven publications stated that the time and effort needed for data collection provides a challenge. In an experience report performed at IONA Technology, Poole and Huisman [S33] mention the high effort of data collection due to them using Excel spreadsheets. Other publications reported that the extra time needed for data collection limited their selection of metrics [S9] [S19]. Another problem that arose through the time and effort needed for data collection was a reduction in data quality, as developers tried to fill in data as quickly as possible [S50]. Furthermore, the time and effort needed for data collection reduced the capability to analyze the development process in real-time due to manual preprocessing steps that had to be performed [S27].

Challenge	Source	# of cases
Data collection 17 (28%)		
Data availability	[S41] [S48] [S49] [S25] [S9] [S19] [S5]	7 (12%)
Time and effort needed for data collection	[S33] [S49] [S9] [S19] [S50] [S51] [S27]	7 (12%)
Data quality	[S50] [S32]	2 (3%)
Data security	[S41]	1 (2%)
Metrics are difficult to implement 14 (23%)		
Difficulty in deriving actionable inputs	[S52] [S19] [S49] [S36]	4 (7%)
Missing alignment between metrics and business objectives	[S52] [S44] [S6]	3 (5%)
Difficult to implement	[S49] [S51]	2 (3%)
Some aspects are hard to measure	[S49] [S39]	2 (3%)
Hard to apply a consistent set of metrics across all phases of product development	[S36] [S8]	2 (3%)
Time and effort needed to measure frequently	[S5]	1 (2%)
Missing understanding 11 (18%)		
Different perceptions of success between stakeholders	[S53] [S42] [S54]	3 (5%)
Mismatch between metric related perceptions among different levels of responsibility	[S55] [S52] [S24]	3 (5%)
Missing understanding for interpretation of metrics	[S52] [S56]	2 (3%)
Lacking support of metrics	[S49] [S33]	2 (3%)
Lack of skills and guidance in setting metrics	[S9]	1 (2%)
Misinterpretation of metrics 6 (10%)		
Subjective or inaccurate information	[S49] [S9] [S57]	3 (5%)
Not enough attention on qualitative aspects of work	[S22] [S58]	2 (3%)
Metrics are easy to manipulate	[S59] [S60]	2 (3%)
Hierarchical boundaries of measurement 6 (10%)		
Fragmented collection and storage of data at different organizational levels	[S61] [S15] [S62] [S43] [S7]	5 (8%)
Feeling of disconnect between team and program level metrics	[S9]	1 (2%)
Coordination challenges in multi-team environment 6 (10%)		
Hard to transfer metrics between teams	[S63] [S64] [S65] [S11] [S42] [S66]	6 (10%)

Table 5.3.: Challenges of metrics in large-scale agile software development

Data quality & data security

The quality of data was identified as a challenge by McMahon [S50] who mentioned that metrics have provided wrong results because developers have put in the wrong data in the measurement tools. Furthermore, Diebold et al. [S32] described the same problem. Here the burndown chart showed fewer finished stories due to developers forgetting to update tasks. Data security has also limited the availability of data, because a governmental company had problems in getting the allowance to collect data for measurement due to privacy reasons [S41].

5.3.2. Metrics are difficult to implement

Another challenge that was frequently mentioned is connected to the implementation of metrics being difficult. Within this class, five categories were identified.

Difficulty in deriving actionable inputs

In their case study, Ram et al. [S19] found that while the data for measurement was available, the company did not use the metric due to having difficulties in deriving value from the metric. This challenge was also stated in the studies from Korkala [S52], Kortum et al. [S49] who both described that the developers did not see any benefits for using the metrics. Another example of the difficulty of deriving actionable inputs is that developers at Atlassian had problems in understanding how their work was linked to the metrix UMUX-Lite [S36].

Missing alignment between metrics and business objectives

Five percent of the identified challenges could be grouped into the category of misalignment between metrics and business objectives. In a case study of twelve Spanish companies using Scrum Garzás and Paulk [S44] identified that 42% of software measurements were not aligned to business objectives which do not follow the best practices for software measurement of the CMMI-DEV 1.3 framework. In a multinational technology company, metrics became misaligned to business objectives because they were adapted to individual interests [S52]. At NYSE Technology, measures were based on profitability which meant that the focus on strategic objectives was lost [S6].

Difficult to implement and some aspects are hard to measure

Kortum et al [S49] and Augustine et al. [S51] mentioned that metrics are hard to implement. The energy company ABB, for example, had difficulty in providing a visualization of task items in form of charts [S51]. Furthermore, Kortum et al. [S49] described the challenge that metrics that can only be calculated by looking at subjective data are hard to measure. This is especially true when trying to measure social conflicts, dysfunctional habits, and social-driven conflicts in software projects [S49]. Furthermore, Alahyari et al. [S39] mentioned that they

didn't have any metrics to measure value, because no clear definition of value existed that could have been measured.

Hard to apply a consistent set of metrics across all phases of product development

Friedmann and Flaouna [S36] [S8] state that ideally the same metrics should be applied across all phases of product development. But this is hard to realize at Atlassin, because, for example, the NPS that is measured only is measurable once the first part of the product is released and used by the customer.

Time and effort needed to measure frequently

In a case study at two Canadian software organizations Mattsson et al. [S5] identified, that although the companies wanted to measure the progress of development daily, measurement was performed every week due to time restrictions.

5.3.3. No shared understanding

Out of the primary studies, eleven citations of challenges related to not having a shared understanding of metrics were identified.

Different perceptions of success between stakeholders

Siddique and Hussein [S53], Fagerholm et al. [S42], and Gregory et al. [S54] all stated that they had problems in introducing metrics due to different perceptions of success between different stakeholders. Siddique and Hussein [S53] describe that customers identify success as the value received in relation to the money invested. Developers define success if the product has high quality, and management identifies it as success when new customers are acquired [S53].

Mismatch between metric-related perceptions among different levels of responsibility.

In a case study at a banking company, Gustavson [S24] identified that the company struggles with mismatched perceptions of metrics among different levels of responsibility. The core team had defined that only sequential dependencies are collected and visualized on the program board. This led to complications during the Scrum of Scrums meeting, as important dependencies were not visualized in the diagram and could not be resolved. Another mismatch of perceptions was described by Chen et al. [S55] between the team and the quality manager and product owner. The team had the perception that faults do not need to be reported in agile software development, while the quality manager and product owner expected all defects to be reported [S55]. Mismatches can furthermore occur when evaluating the effectiveness of metrics. Korkala [S52] reports, that high-level management thought that the implemented metrics provide are useful, while product line management and quality assurance are not convinced.

Missing understanding for interpretation of metrics

In order to receive the optimal value of metrics they must be understood [S52]. Korkala [S52] and Pacheco et al. [S56], both describe that understanding how metrics can be interpreted is perceived challenging. Understanding variation in a graph showing the relation between size and effort will provide more information if the reasons behind the variations are understood [S52].

Lacking support of metrics

IONA Technology stated the challenge that, while metrics would be very beneficial during re-planning activities, they are not used because of the lacking support of developers [S33]. This is supported by the survey of Kortum et al. [S49] who identify the lack of support of metrics both in team and management.

Lack of skills and guidance in setting metrics

Korpivaara et al. [S9] mention that there is a lack of skills and guidance for setting metrics in the studied company. Good management decisions based on metrics were proven difficult due to a lack of standardized measurement practices within the company [S9].

5.3.4. Misinterpretation of metrics

Not enough focus on qualitative aspects of work

Matthiesen and Bjorn [S22] identified in a case study that because of the focus of metrics like the burndown chart, the focus on qualitative aspects of work got lost. This means that, while focusing on the speed of development, team collaboration and quality of work can fade into the background [S22]. Microsoft reports the same issue with the measure of the velocity of task completion [S58]. While being a good measure of identifying whether the requirements were delivered within an iteration, velocity is not enough to identify the cause of variations in the process [S58].

Subjective and inaccurate information

Subjective and inaccurate information form measurement challenges [S9] [S49]. A large banking company had the challenge that the team perceived that external factors influenced the outcomes of the measurements [S9]. The resulting lack of influence they, therefore, had on the outcome of the metric led to the team losing trust in the measurement [S9]. This issue is also reported in a case study of Hassani-Alaoui et al. [S57] who reported that a company measured success of an project by being on-time, on-budged and high-quality. With these measures, the problem arose that deviations from budget and time are often related to wrong estimation instead of wrong management [S57].

Metrics are easy to manipulate

Another challenge of metrics is that they are easy to manipulate [S59] [S60]. Hekkala et al. [S59] describe this problem in relation to the focus on story points implemented. When the amount of story points is used for evaluating the performance of the sprint, the team might be led to agreeing on more issues than they can actually develop or adapt the estimation [S59]. The same challenge was mentioned in an experience report of Elssamadisy and Schalliol [S60] who describe the misuse of the velocity metric. Here, developers were asked to reduce the effort for testing and refactoring in order to retain their velocity [S60]. This led to a reduction of quality and finally to an increase in project time [S60].

5.3.5. Hierarchical boundaries of measurement

Fragmented collection and storage of data at different organizational levels

Fragmented collection and storage of data was identified as a challenge by five publications. Lehto and Rautiainen [S62] identified the problem that people on different organizational levels were using various spreadsheets for managing backlog items. As those spreadsheets were not linked with each other in real-time, data inconsistencies occurred. [S62] Furthermore, in Fannie Mae [S15] metrics could not be aligned at the enterprise level, as tools were implemented separately from each other across different products and could not be merged. This was also mentioned in a case study by Fabjan et al. [S61] who found that while information exists it is not shared on all levels due to data being stored at different places which can not be accessed by everyone.

Feeling of disconnect between team and program level metrics

In a large Nordic bank [S9] team members reported a feeling of disconnect between the portfolio level metrics and the team. Due to the metric being aggregated on a high level, the team felt that their actions did not influence the metrics at all.

5.3.6. Coordination challenges in multi-team environment

Hard to transfer metrics between teams

In six cases, publications stated that transferring metrics between teams is challenging. One of the reasons for this was the problem that different teams use different effort estimates and therefore measurements are not comparable [S63] [S11] [S64] [S52] [S42][S1]. Salameh et al. [S65] mentioned that measurements could not be compared across teams, because teams had individual metrics implemented.

5.4. Success factors of using and implementing metrics in large-scale agile software development

What success factors of using and implementing metrics in large-scale agile software development have been reported?

Table 5.4 displays the 25 success factors that were mentioned in the primary studies. These 25 success factors were grouped into ten categories. The most mentioned success factor categories are providing accessibility (18%) and usage of tools (18%). Ten percent of cases suggest an iterative operationalization of metrics is important for successfully starting a measurement program. Additionally, teams and stakeholders should be included into the measurement process (10%). Furthermore, the primary studies suggested that metrics should be used over all organizational levels (10%) and trust and cooperation of teams towards metrics needs to be ensured (10%). Additionally, alignment of metrics with objectives, processes, and available data (9%), keeping it simple by not introducing too many metrics (7%), horizontal alignment of metrics (5%), and defining interventions that are linked to KPIs (3%).

Success factors of metrics

Success Factor	Source	# of cases
Accessibility 12 (18%)		
Make metrics visible to everyone	[S14] [S67] [S68] [S28] [S9] [S35] [S52] [S49]	8 (12%)
Frequent monitoring of metrics	[S68] [S50] [S9] [S10]	4 (6%)
Alignment of metrics 6 (9%)		
Alignment of metrics with objectives	[S44] [S43]	2 (3%)
Metrics should be defined by looking at available data	[S41] [S10]	2 (3%)
Alignment of metrics with the development process	[S41] [S10]	2 (3%)
Horizontal alignment 4 (6%)		
Have the same metrics across multiple projects	[S15] [S10] [S35]	3 (4%)
Monitor the same metrics throughout all phases	[S8]	1 (1%)
Interpretation of metrics 2 (3%)		
Improve visibility instead of focusing on deviations	[S69]	1 (1%)
Define interventions that are linked to KPIs	[S68]	1 (1%)
Involve team and stakeholders 7 (10%)		
Metrics should be set up around teams and people	[S68] [S10] [S31]	3 (4%)
Include teams in defining objectives and measures	[S42] [S6] [S14]	3 (4%)
Allow customization of metrics	[S70]	1 (1%)
Iterative operationalization 7 (10%)		
Introduce metrics in an iterative fashion	[S41] [S71] [S52] [S72] [S15]	5 (7%)
Regular feedback about metric implementation and interpretation	[S41] [S9]	2 (3%)

Continuation of table

Success Factor	Source	# of cases
Keep it simple 5 (7%)		
Don't use too many metrics	[S25] [S52] [S35]	3 (4%)
Perform a cost-benefit analysis	[S73] [S10]	2 (3%)
Metrics used over all organizational levels 7 (10%)		
Define metrics at all organizational levels	[S66] [S42] [S52] [S6]	4 (6%)
Standardize reporting management systems across all organizational levels	[S14] [S74] [S66]	3 (4%)
Use tools 12 (18%)		
Automate data collection and calculation of metrics by using tools	[S12] [S73] [S75] [S71] [S25] [S14] [S35] [S76] [S17] [S10] [S51] [S41]	12 (18%)
Trust and cooperation of teams 7 (10%)		
Build transparency	[S41] [S28] [S52]	3 (4%)
Ensure commitment of management and teams	[S51] [S10]	2 (3%)
Ensure that the team trusts the measurement program	[S27] [S41]	2 (3%)

Table 5.4.: Success factors of metrics in large-scale agile software development

5.4.1. Accessibility

Accessibility of metrics was the most frequently mentioned success factor (18%) within the primary sources.

Make metrics visible to everyone

Overall, eight primary studies identified making metrics visible to everyone as a success factor. Korpivara et al. [S9] found that providing visibility between team and unit metrics is crucial. Integraph, a software developer with more than 75 teams, observed that sharing all measurements with everyone in the organization increased the overall acceptance of metrics [S14]. A good approach for providing this kind of visibility is the usage of dashboards [S49], which can, for example, be used to provide an overview of all indicators within the organization [S52]. Additionally, making metrics accessible to everyone in the organization increases transparency and improves planning [S68].

Frequent monitoring of metrics

Three publications identified frequent monitoring of KPIs as a success factor of using metrics in large-scale agile software development. Cheng et al. [S68] found that monitoring the KPIs in the daily Scrum meeting made timely interventions possible. McMahon et al. [S50] identified in a case study that the likelihood of the measurements being used for improvement increased

with the recency of measurements. Additionally, frequent monitoring of measurement results was found to correlate with team motivation [S9] positively.

5.4.2. Alignment of metrics

Alignment of metrics with objectives

Following the best practice of CMMI-DEV 1.3 [S44], the alignment of measurements with objectives was identified as a success factor by three publications. According to [S43] measurements should match with the agile goals that were identified. Horlach et al. [S6] mentioned that the alignment of measurement and goals increased awareness and enabled commitment.

Define metrics by looking at available data

Ram et al. [S41] and Staron and Meding [S10] stated that the implementation of metrics should be aligned with the available data. In order to keep the effort for data collection minimal, the introduction of metrics should start with using already existing data [S10] [S41]. After successful implementation of those metrics, the measurement process can be refined, and additional metrics can be introduced [S10] [S41].

Align metrics with the development process

In addition to looking at available data, measurements should be aligned with the development process to reduce additional overhead [S41] [S10]. Integrating measurements into the development process using existing tools for data acquisition increased the acceptance of measurement practices because no additional effort for learning was needed [S10].

5.4.3. Horizontal alignment

The same metrics should be used across all projects to enable the possibility of comparing different projects with each other [S15]. Meding [S35] stated that the same dashboards should be used across multiple teams. Furthermore, Flaounas and Firedman [S8] [S36] state that ideally, the same metrics should be used throughout the whole development process.

5.4.4. Interpretation of metrics

For the interpretation of metrics, two success factors were identified. In an experience report, Sirkiä and Laanti [S69] found that metrics should be used to provide visibility and give information instead of focusing on control. Instead of explaining variances to the management, measures like estimation accuracy are used to help the team in planning [S69]. Cheng et al. [S68] state that all KPIs should be linked to interventions that are automatically triggered once the defined target of the KPI is undercut. When the measure of productivity on the team level, for example, went below a threshold, other teams were asked to help

the struggling team with the task [S68]. Furthermore, meetings were held with the team to identify possible problems, or tasks were removed from the sprint backlog [S68].

5.4.5. Involve team and stakeholders

Metrics should be set up around teams and stakeholders

Measurements should be set up around teams since they are responsible for developing the software and are therefore the most important aspect to monitor [S68]. In a case study, Staron and Meding [S10] identified that each measurement system in the company should be linked to a stakeholder. Otherwise, this can lead to measurements not being used [S10]. A further aspect of linking measurements around people is described by Janus et al. [S31] who concluded that quality managers should be part of the development team. This ensures correct interpretation of measurement and increases the usefulness of measurements [S31].

Include the team in defining objectives and measures

Involving team members in defining objectives and measurements has been named as a success factor by three publications. At Integraph, all levels of the organization are included in the definition of objectives and measurements so that everyone can make decisions based on the same information [S14]. In a case study, Fagerholm et al. [S42] identified that involving teams in goal setting is perceived as a fundamental characteristic of high-performing teams. Furthermore, Horlach et al. [S6] identified that on the team level, measurements and objectives should be self-defined to increase the understanding of purpose in the team. Those objectives should be derived from the overall business objectives defined at higher levels [S6].

Allow customization of metrics

Allowing customization is the last success factor that could be identified in the category of involving team and stakeholders. In an action research approach at a European Healthcare provider, Colye and Barata [S70] identified that changing metrics for each team or project in an agile way improved team commitment [S70].

5.4.6. Iterative operationalization

Introduce metrics in an iterative fashion

Similar to the agile development process, measurements should be implemented in an iterative fashion [S41]. Even if the company wants to implement multiple metrics, one should first start with choosing a few key measurements. After these are successfully implemented and used by developers further metrics can be introduced [S41] [S52] [S72]. Horhonen [S71] describes this approach with an example of a defect reporting process. First, guidelines regarding which defects should be reported are defined. Then, the appropriateness of the defect reporting tools and processes is assessed. After a process is established, appropriate metrics for defect reporting are selected and implemented. [S71]

Regular feedback about metric implementation and interpretation

Not only should metrics be introduced in an iterative fashion but also feedback about metric implementation and interpretation should be provided in that way. Ram et al. [S41] highlights the importance of frequent reevaluation of measurements during metric implementation. Feedback sessions between development teams and the measurement team enabled reprioritization and adaption of metrics, as well as learning, [S41]. Korpivaara [S9] identified that, especially on the program level, frequent review and reprioritization of metrics is necessary since the importance of metrics changed with the maturity level of agile practices.

5.4.7. Keep it simple

Do not use too many metrics

Three researchers stated the fact that organizations should not use too many metrics. While collecting multiple metrics is important, the number of metrics should not be too high, and only essential metrics should be implemented to keep the time needed for measurement to a minimum [S52] [S35] [S25].

Perform a cost-benefit analysis

In order to avoid introducing unhelpful metrics into the development process, two authors mentioned that a cost-benefit analysis should be performed [S73] [S10]. Furthermore, the cost for measurement introduction can be reduced by reusing base measures that have been collected, specified, and stored in a catalog [S10].

5.4.8. Metrics used over all organizational levels

Define metrics at all organizational levels

Four publications identified defining metrics and objectives at all organizational levels as an important success factor of measurement. Fagerholm et al. [S42] state that the measurement of large organizations should start by understanding objectives and measurement needs at all levels of the organization and aligning those needs. In introducing a quality governance program, Pradhan and Nanniyur [S66] defined metrics at all organizational levels to make sure that quality is a decision-making criterion for everyone. Horlach et al. [S6] aligned the synchronization cycles of all organizational levels within the company.

Standardize reporting management systems across all organizational levels

With large-scale agile software development, the need to standardize reporting management systems across all organizational levels arises, as three publications report. At Integraph, a large software development company, the same reports are used at all organizational levels to ensure that everyone has the same information [S14]. This means that on the portfolio level, the measurements are also performed based on the measured velocity at the team

level [S14]. Moreover, Winz et al. [S74] state that standardized measurement systems should ensure that all teams follow the same best practices in reporting. This assures correct and efficient reporting, as data is collected in the right way from the beginning and does not have to be reprocessed [S74]. In order to achieve standardized reporting management systems, data must be integrated company-wide [S66]. Only this ensures that all measurements are included in the decision-making process [S66].

5.4.9. Use tools

In twelve cases, the automatization of data collection and metric calculation through the usage of tools has been mentioned as a success factor. Multiple publications state that tools should be used in order to reduce the time needed for data collection [S73] [S25] [S10] [S12] [S14]. Stettina and Schoemaker [S12] identified that organizations with higher agile maturity had a higher, more frequently used automated data collection instead of using spreadsheets. This is in line with Sutherland et al. [S17], who reported that high-velocity projects need automated measurements to track the development status across teams. Tekin et al. [S75] mention that ideally, team members should be included in the decision of which tools for measurement are used to decide about the usability and the functionality of the tools.

5.4.10. Trust and cooperation of teams

Ensure commitment of management and teams

Ensuring management and team commitment towards measurements has been identified as a success factor by two studies. Staron and Meding [S10] group commitment into two categories: management commitment and team commitment. Management commitment was needed to ensure that the measurement program was actually introduced. This was achieved by sharing predictions of measurements [S10]. Ensuring team commitment was deemed critical to ensure the reliability of data that is entered by the team [S10]. Augustine et al. [S51] ensured commitment by sharing success stories of metric usage.

Ensure that the team trusts the measures

Furthermore, ensuring the trust of teams was perceived as a success factor of metric usage. In order to ensure the collection of accurate data, the team needs to have trust that metrics are not used to their disadvantage [S27]. Moreover, trust in the accuracy of metrics should be established by using familiar metrics or calculating the trustworthiness of metrics [S41].

Build transparency

Another success factor in achieving the trust and cooperation of teams is to achieve transparency [S41] [S28] [S52]. Ram et al. [S41] achieved this by linking metrics to the actual data source and providing everyone visibility of how the metric is calculated. Tably et al. [S28]

presented the data of measurements to the whole team and openly discussed the reasons for the implementation of each measurement.

6. Discussion

In this chapter, the key findings of this master's thesis will be presented. Furthermore, the limitations of the SLR will be discussed.

6.1. Key findings

Increasing number of publications on metrics in large-scale agile software development

There is an increasing interest of researchers for metrics in large-scale agile software development. This can be concluded from the increasing number of studies being published in this field of research by year. This is in line with the growing number of agile software development being introduced within large organizations [2].

Most studies are published in Conferences

With 83 of 110 studies, most primary studies were published in conferences. From this, it can be concluded that the research field of metrics in large-scale agile software development has not achieved high maturity yet, since conference publications have a lower maturity than journal publications. This conclusion is also supported by the distribution of research approaches throughout the primary studies. At 34%, experience reports represent the second-largest share of the identified research approaches. Most studies related to metrics in large-scale agile software development were published in the Agile Conference (18 publications) and the Hawaii International Conference on System Sciences (13 publications). The most frequently represented journal was IEEE Software (4 publications), and the most frequently represented workshop the ISCE workshop on software development governance (3 publications).

Most measurements are conducted at the team level

Most measurements described in literature are conducted at the team level (69%). These include measures of velocity, the number of identified defects, or estimation accuracy. Measurements on the program level are the second most frequently found metrics (26%). With the number of identified defects, the number of tests, and estimation accuracy, the measurements on the program level are very similar to the measurements conducted at the team level. While many publications described that they are using metrics at the portfolio level only 22 metrics at the portfolio level were actually described. Portfolio level measures are mostly connected to economic entities or customer satisfaction. These findings align with the metrics described in the SAFE framework [61].

Most measurements conducted were process metrics

Similar to the findings in the SLR of Kupiainen et al. [5], process and product were the most measured entities. Within processes, measurements were mostly applied during testing, implementation, the whole development process, and estimation. In line with the agile value *"Working software over comprehensive documentation"* [8] only few measures regarding constructing specifications and requirement engineering were found. The most identified product metrics were measures of customer satisfaction or customer value delivered.

Data collection and storage is the most common challenge of using and implementing metrics in large-scale agile software development

In the SLR, 21 challenges, grouped into six categories: using and implementing metrics in large-scale agile software development were identified. The most frequently mentioned challenge categories were data collection (28%), metrics being difficult to implement (23%), and no shared understanding of metrics (18%). The five most frequently mentioned challenges were data availability (12%), time and effort needed for data collection (12%), metrics being hard to transfer between teams (10%), fragmented collection and storage of data at different organizational levels (8%), and difficulty of deriving actionable inputs (7%).
In large-scale agile software development, the acquisition and processing of data seem to provide the biggest challenge, since three out of the five most mentioned challenges were related to data collection and storage of data. Furthermore, the large-scale setting where multiple teams are working on one product introduces the challenge of comparing teams.
To our knowledge, there is not much scientific literature about challenges in the introduction and usage of metrics in large-scale agile software development. Within the Version One State of Agile surveys [12], fragmented tooling and project-related data/measurements have been stated as a general challenge when adopting and scaling agile. This is in line with the findings in this thesis. The results of this research question are in line with the challenges that Ram et al. [S19] identified. In a case study where metrics were introduced in four companies lack of data or tools, existing processes inhibiting change, and the difficulty of deriving actionable inputs were found to be the main challenges.

Automated data collection, making metrics visible to everyone, and introducing metrics in an iterative fashion are the most identified success factors of metric usage and implementation

In this literature review, 22 success factors and ten success factor categories have been identified. The most frequently mentioned success factor categories were providing accessibility (18%), using tools to automate data collection (18%), and involving team and stakeholders (10%). The most common mentioned success factors are automated data collection (18%), making metrics visible to everyone (12%), and introducing metrics in an iterative fashion (7%). Within the Version One State of Agile surveys [**version**], five tips for successfully scaling agile software development have been mentioned. The second most frequently agreed-on success factor being using a common tool across all teams (40%) [2]. This is in line with the success

factor by standardizing reporting management across all organizational levels, projects, and teams that have been identified in this thesis. When introducing a software measurement program in an industrial project, Poligadu and Moloo [62] provided an overview of multiple success factors for measurement program implementation in agile development. These include the alignment of measures to agile principles, providing trend information, aligning the metrics to processes and available data, and keeping it simple by not using too many metrics [62]. The aforementioned success factors align very well with the success factors identified within this master's thesis.

6.2. Limitations

This section describes the limitations that might have occurred within this thesis. First, the results of this SLR can be affected by the completeness of the identified primary studies. To overcome this problem, this SLR followed the recommendations of Zhang et al. [16]. By performing a preliminary search, the keywords could be adapted and optimized. Furthermore, the comparison of the main search results with previous literature reviews and the preliminary search enabled a validity check of the automated search. Nonetheless, these approaches do not guarantee that the search terms missed no primary studies.
Another aspect that might have affected the completeness of identified primary studies is researcher bias. Misinterpretation of inclusion and exclusion criteria or human error in the filtering process might have affected the completeness. This risk was mitigated by having a second researcher perform the filtering of metadata and resolving conflicts. A second researcher applied the inclusion and exclusion criteria to random samples during the abstract and full-text filtering in regular intervals. Afterward, conflicts were discussed and resolved together.
Another part that might have been influenced by completeness and bias is the data extraction and coding process. These steps were performed in multiple iterations to mitigate the bias and completeness of data extraction and coding. Furthermore, we try to make the coding process traceable by linking each result to the primary study it was identified in.

7. Conclusion

This chapter summarizes the results of this master thesis and provides an overview about possibilities for future work.

7.1. Summary

In the following section, the answers to the research questions presented in Section 1.2 are summarized. This master's thesis provides an overview about the current state of research on goals and metrics in large-scale agile software development. By performing a SLR, a set of 17042 publications that was retrieved from the keyword search was analyzed, resulting in 110 primary studies about goals and metrics in large-scale agile software development. The growing number of publications in this research field shows that there is increasing attention from researchers. Due to the large number of the identified studies being published in conferences, it can be concluded that the research area is still at a low maturity.

> **Research Question 1:** *What are goals and metrics in large-scale agile software development?*

Within the primary studies, increasing quality, operating at optimal capacity, and increasing customer satisfaction were the most frequently mentioned goals. These goals align very well with the metrics that were identified in this SLR. Metrics in large-scale agile software development are most frequently measured on the team level and the program level. Metrics used on those two levels are very similar. While multiple organizations described that they were using metrics on the portfolio level, only 20 metrics were described.

> **Research Question 2:** *What challenges of using and implementing metrics in large-scale agile software development have been reported?*

Collection and storage of data is the most mentioned challenge of using and implementing metrics in large-scale agile software development. This includes data availability, time and effort needed for data collection, and fragmented collection and storage of data at different organizational levels. Further reported challenges included metrics being difficult to implement and the lack of shared understanding of metrics and measurement goals.

> **Research Question 3:** *What success factors of using and implementing metrics in large-scale agile software development have been reported?*

Within the 110 primary studies, 22 success factors grouped into ten categories were identified. Automated data collection, making metrics visible to everyone, and introducing metrics in an iterative fashion are the most identified success factors of metric usage and implementation in large-scale agile software development.

7.2. Future work

The ideas for future work are derived from the findings in this master's thesis. As described in the previous chapters, even though the usage of metrics at the portfolio level was mentioned in multiple cases, only a few metrics were actually described. Performing case studies on this field of research could therefore be a good point for future work.
Furthermore, research on resolving the challenges described in this master thesis would be a good starting point for future research. This could include the creation of a KPI catalog to provide a structured overview about metrics, their calculation rules, data needed, and objectives they can be linked to. This could help solve the challenge of deriving actionable inputs of measurements or having misalignment between metrics and business objectives. Another good starting point lies in for future work lies in analyzing the tools and processes used for measurements in large-scale agile software development. This would be useful to provide opportunities to overcome the challenge of fragmented collection and storage of data at different organizational levels and is in line with the success factor of using tools to automate data collection and calculation of metrics.

A. Appendix

A.1. Search terms of the main search

Search terms of the main search

DB	Search term
ACM	Abstract:(((("agile" OR "agility" OR "extreme programming" OR "XP" OR "feature driven development" OR "FDD" OR "scrum" OR "crystal" OR "pair programming" OR "test-driven development" OR "TDD" OR "leanness" OR "lean software development" OR "lean development" OR "LSD") AND NOT "manufacturing") OR ((("agile" OR "agility" OR "extreme programming" OR "XP" OR "feature driven development" OR "FDD" OR "scrum" OR "crystal" OR "pair programming" OR "test-driven development" OR "TDD" OR "leanness" OR "lean software development" OR "lean development" OR "LSD") AND NOT "manufacturing") AND ("large scale" OR "scaling")) OR ("Crystal Family" OR "Dynamic Systems Development Method Agile Project Framework for Scrum" OR "Scrum of Scrums" OR "Enterprise Scrum" OR "Agile Software Solution Framework" OR "Large-Scale Scrum" OR "Scaled Agile Framework" OR "Disciplined Agile" OR "Spotify Model" OR "Mega Framework" OR "Enterprise Agile Delivery and Agile Governance Practice" OR "Recipes for Agile Governance in the Enterprise" OR "Continuous Agile Framework" OR "Scrum at Scale" OR "Enterprise Transition Framework" OR "ScALeD Agile Lean Development" OR "eXponential Simple Continuous Autonomous Learning Ecosystem" OR "Lean Enterprise Agile Framework" OR "Nexus" OR "FAST Agile")) AND ((measur* ORmetric* OR "diagnostic" OR monitor* OR quantitative OR "reporting" OR indicator* OR "performance indicator" OR "KPI") OR (aim OR target OR object* OR goal)))) AND Keyword:(((("agile" OR "agility" OR "extreme programming" OR "XP" OR "feature driven development" OR "FDD" OR "scrum" OR "crystal" OR "pair programming" OR "test-driven development" OR "TDD" OR "leanness" OR "lean software development" OR "lean development" OR "LSD") AND NOT "manufacturing") OR ((("agile" OR "agility" OR "extreme programming" OR "XP" OR "feature driven development" OR "FDD" OR "scrum" OR "crystal" OR "pair programming" OR "test-driven development" OR "TDD" OR "leanness" OR "lean software development" OR "lean development" OR "LSD") AND NOT "manufacturing") AND ("large scale" OR "scaling")) OR ("Crystal Family" OR "Dynamic Systems Development Method Agile Project Framework for Scrum" OR "Scrum of Scrums" OR "Enterprise Scrum" OR "Agile Software Solution Framework" OR "Large-Scale Scrum" OR "Scaled Agile Framework" OR "Disciplined Agile" OR "Spotify Model" OR "Mega Framework" OR "Enterprise Agile Delivery and Agile Governance Practice" OR "Recipes for Agile Governance in the Enterprise" OR "Continuous Agile Framework" OR "Scrum at Scale" OR "Enterprise Transition Framework" OR "ScALeD Agile Lean Development" OR "eXponential Simple Continuous Autonomous Learning Ecosystem" OR "Lean Enterprise Agile Framework" OR "Nexus" OR "FAST Agile")) AND ((measur* ORmetric* OR "diagnostic" OR monitor* OR quantitative OR "reporting" OR indicator* OR "performance indicator" OR "KPI") OR (aim OR target OR object* OR goal)))) AND Title:(((("agile" OR "agility" OR "extreme programming" OR "XP" OR "feature driven development" OR "FDD" OR "scrum" OR "crystal" OR "pair programming" OR "test-driven development"

Continuation of Table A.1.: Search terms of main search

DB	Search term
	OR "TDD" OR "leanness" OR "lean software development" OR "lean development" OR "LSD") AND NOT "manufacturing") OR ((("agile" OR "agility" OR "extreme programming" OR "XP" OR "feature driven development" OR "FDD" OR "scrum" OR "crystal" OR "pair programming" OR "test-driven development" OR "TDD" OR "leanness" OR "lean software development" OR "lean development" OR "LSD") AND NOT "manufacturing") AND ("large scale" OR "scaling")) OR ("Crystal Family" OR "Dynamic Systems Development Method Agile Project Framework for Scrum" OR "Scrum of Scrums" OR "Enterprise Scrum" OR "Agile Software Solution Framework" OR "Large-Scale Scrum" OR "Scaled Agile Framework" OR "Disciplined Agile" OR "Spotify Model" OR "Mega Framework" OR "Enterprise Agile Delivery and Agile Governance Practice" OR "Recipes for Agile Governance in the Enterprise" OR "Continuous Agile Framework" OR "Scrum at Scale" OR "Enterprise Transition Framework" OR "ScALeD Agile Lean Development" OR "eXponential Simple Continuous Autonomous Learning Ecosystem" OR "Lean Enterprise Agile Framework" OR "Nexus" OR "FAST Agile")) AND ((measur* OR-metric* OR "diagnostic" OR monitor* OR quantitative OR "reporting" OR indicator* OR "performance indicator" OR "KPI") OR (aim OR target OR object* OR goal))))

Continuation of Table A.1.: Search terms of main search

DB	Search term
AISLE	abstract:(((("agile" OR "agility" OR "extreme programming" OR "XP" OR "feature driven development" OR "FDD" OR "scrum" OR "crystal" OR "pair programming" OR "test-driven development" OR "TDD" OR "leanness" OR "lean software development" OR "lean development" OR "LSD") AND NOT "manufacturing") OR ((("agile" OR "agility" OR "extreme programming" OR "XP" OR "feature driven development" OR "FDD" OR "scrum" OR "crystal" OR "pair programming" OR "test-driven development" OR "TDD" OR "leanness" OR "lean software development" OR "lean development" OR "LSD") AND NOT "manufacturing") AND ("large scale" OR "scaling")) OR ("Crystal Family" OR "Dynamic Systems Development Method Agile Project Framework for Scrum" OR "Scrum of Scrums" OR "Enterprise Scrum" OR "Agile Software Solution Framework" OR "Large-Scale Scrum" OR "Scaled Agile Framework" OR "Disciplined Agile" OR "Spotify Model" OR "Mega Framework" OR "Enterprise Agile Delivery and Agile Governance Practice" OR "Recipes for Agile Governance in the Enterprise" OR "Continuous Agile Framework" OR "Scrum at Scale" OR "Enterprise Transition Framework" OR "ScALeD Agile Lean Development" OR "eXponential Simple Continuous Autonomous Learning Ecosystem" OR "Lean Enterprise Agile Framework" OR "Nexus" OR "FAST Agile")) AND ((measur* OR metric* OR "diagnostic" OR monitor* OR "quantitative" OR indicator* OR "performance indicator" OR "KPI") OR (aim OR target OR object* OR goal))) OR title:(((("agile" OR "agility" OR "extreme programming" OR "XP" OR "feature driven development" OR "FDD" OR "scrum" OR "crystal" OR "pair programming" OR "test-driven development" OR "TDD" OR "leanness" OR "lean software development" OR "lean development" OR "LSD") AND NOT "manufacturing") OR ((("agile" OR "agility" OR "extreme programming" OR "XP" OR "feature driven development" OR "FDD" OR "scrum" OR "crystal" OR "pair programming" OR "test-driven development" OR "TDD" OR "leanness" OR "lean software development" OR "lean development" OR "LSD") AND NOT "manufacturing") AND ("large scale" OR "scaling")) OR ("Crystal Family" OR "Dynamic Systems Development Method Agile Project Framework for Scrum" OR "Scrum of Scrums" OR "Enterprise Scrum" OR "Agile Software Solution Framework" OR "Large-Scale Scrum" OR "Scaled Agile Framework" OR "Disciplined Agile" OR "Spotify Model" OR "Mega Framework" OR "Enterprise Agile Delivery and Agile Governance Practice" OR "Recipes for Agile Governance in the Enterprise" OR "Continuous Agile Framework" OR "Scrum at Scale" OR "Enterprise Transition Framework" OR "ScALeD Agile Lean Development" OR "eXponential Simple Continuous Autonomous Learning Ecosystem" OR "Lean Enterprise Agile Framework" OR "Nexus" OR "FAST Agile")) AND ((measur* OR metric* OR "diagnostic" OR monitor* OR "quantitative" OR indicator* OR "performance indicator" OR "KPI") OR (aim OR target OR object* OR goal)))

Continuation of Table A.1.: Search terms of main search

DB	Search term
Scopus	(((((((((TITLE-ABS ("large scale") OR TITLE-ABS ("scaling")) AND ((TITLE-ABS ("agile") OR TITLE-ABS ("agility") OR TITLE-ABS ("extreme programming") OR TITLE-ABS (" XP") OR TITLE-ABS ("feature driven development") OR TITLE-ABS ("FDD") OR TITLE-ABS ("scrum") OR TITLE-ABS ("crystal") OR TITLE-ABS ("pair programming") OR TITLE-ABS ("test-driven development") OR TITLE-ABS ("TDD") OR TITLE-ABS (" leanness") OR TITLE-ABS ("lean software development") OR TITLE-ABS ("lean development") OR TITLE-ABS ("LSD")) AND NOT (TITLE-ABS ("manufacturing")))) OR (TITLE-ABS ("Crystal Family") OR TITLE-ABS ("Dynamic Systems Development Method Agile Project Framework for Scrum") OR TITLE-ABS ("Scrum-of-Scrums") OR TITLE-ABS ("Enterprise Scrum") OR TITLE-ABS ("Agile Software Solution Framework") OR TITLE-ABS ("Large Scale Scrum") OR TITLE-ABS ("Scaled Agile Framework") OR TITLE-ABS ("Disciplined Agile 2.0") OR TITLE-ABS ("Spotify Model") OR TITLE-ABS ("Mega Framework") OR TITLE-ABS ("Enterprise Agile Delivery and Agile Governance Practice") OR TITLE-ABS ("Recipes for Agile Governance in the Enterprise") OR TITLE-ABS ("Continuous Agile Framework") OR TITLE-ABS ("Scrum at Scale") OR TITLE-ABS ("Enterprise Transition Framework") OR TITLE-ABS ("ScALeD Agile Lean Development") OR TITLE-ABS ("eXponential Simple Continuous Autonomous Learning Ecosystem") OR TITLE-ABS ("Lean Enterprise Agile Framework") OR TITLE-ABS ("Nexus") OR TITLE-ABS ("FAST Agile")) OR ((TITLE-ABS ("agile") OR TITLE-ABS ("agility") OR TITLE-ABS ("extreme programming") OR TITLE-ABS (" XP") OR TITLE-ABS ("feature driven development") OR TITLE-ABS ("FDD") OR TITLE-ABS ("scrum") OR TITLE-ABS ("crystal") OR TITLE-ABS ("pair programming") OR TITLE-ABS ("test-driven development") OR TITLE-ABS ("TDD") OR TITLE-ABS (" leanness") OR TITLE-ABS ("lean software development") OR TITLE-ABS ("lean development") OR TITLE-ABS ("LSD")) AND NOT (TITLE-ABS ("manufacturing")))) AND (TITLE-ABS-KEY (measur*) OR TITLE-ABS-KEY (metric*) OR TITLE-ABS-KEY ("diagnostic") OR TITLE-ABS-KEY (monitor*) OR TITLE-ABS-KEY ("quantitative") OR TITLE-ABS-KEY (indicator*) OR TITLE-ABS-KEY ("performance indicator") OR TITLE-ABS-KEY ("reporting") OR TITLE-ABS-KEY ("KPI") OR KEY ("aim") OR TITLE ("aim") OR KEY ("target") OR TITLE ("target") OR KEY (object*) OR TITLE (object*) OR KEY ("goal") OR TITLE ("goal"))))))))) AND (LIMIT-TO (SUBJAREA , "COMP") OR LIMIT-TO (SUBJAREA , "ENGI") OR LIMIT-TO (SUBJAREA , "MATH") OR LIMIT-TO (SUBJAREA , "MULT") OR LIMIT-TO (SUBJAREA , "BUSI") OR LIMIT-TO (SUBJAREA , "ECON") OR LIMIT-TO (SUBJAREA , "Undefined") OR EXCLUDE (SUBJAREA , "MATE") OR EXCLUDE (SUBJAREA , "PHYS") OR EXCLUDE (SUBJAREA , "CHEM") OR EXCLUDE (SUBJAREA , "CENG") OR EXCLUDE (SUBJAREA , "ENER") OR EXCLUDE (SUBJAREA , "BIOC") OR EXCLUDE (SUBJAREA , "ENVI") OR EXCLUDE (SUBJAREA , "EART") OR EXCLUDE (SUBJAREA , "MEDI") OR EXCLUDE (SUBJAREA , "AGRI") OR EXCLUDE (SUBJAREA , "PHAR")

Continuation of Table A.1.: Search terms of main search

DB	Search term
	OR EXCLUDE (SUBJAREA , "ARTS") OR EXCLUDE (SUBJAREA , "HEAL") OR EXCLUDE (SUBJAREA , "DENT") OR EXCLUDE (SUBJAREA , "IMMU") OR EXCLUDE (SUBJAREA , "NEUR") OR EXCLUDE (SUBJAREA , "NURS") OR EXCLUDE (SUBJAREA , "VETE")) AND (LIMIT-TO (DOCTYPE , "cp") OR LIMIT-TO (DOCTYPE , "ar") OR LIMIT-TO (DOCTYPE , "dp")) AND (LIMIT-TO (PUBYEAR , 2021) OR LIMIT-TO (PUBYEAR , 2020) OR LIMIT-TO (PUBYEAR , 2019) OR LIMIT-TO (PUBYEAR , 2018) OR LIMIT-TO (PUBYEAR , 2017) OR LIMIT-TO (PUBYEAR , 2016) OR LIMIT-TO (PUBYEAR , 2015) OR LIMIT-TO (PUBYEAR , 2014) OR LIMIT-TO (PUBYEAR , 2013) OR LIMIT-TO (PUBYEAR , 2012) OR LIMIT-TO (PUBYEAR , 2011) OR LIMIT-TO (PUBYEAR , 2010) OR LIMIT-TO (PUBYEAR , 2009) OR LIMIT-TO (PUBYEAR , 2008) OR LIMIT-TO (PUBYEAR , 2007) OR LIMIT-TO (PUBYEAR , 2006) OR LIMIT-TO (PUBYEAR , 2005) OR LIMIT-TO (PUBYEAR , 2004) OR LIMIT-TO (PUBYEAR , 2003) OR LIMIT-TO (PUBYEAR , 2002) OR LIMIT-TO (PUBYEAR , 2001)) AND (EXCLUDE (EXACTSRCTITLE , "Advanced Materials Research") OR EXCLUDE (EXACTSRCTITLE , "Applied Mechanics And Materials") OR EXCLUDE (EXACTSRCTITLE , "Conference Record Of The IEEE Photovoltaic Specialists Conference") OR EXCLUDE (EXACTSRCTITLE , "Energy And Buildings") OR EXCLUDE (EXACTSRCTITLE , "2013 Conference On Lasers And Electro Optics Europe And International Quantum Electronics Conference CLEO Europe IQEC 2013") OR EXCLUDE (EXACTSRCTITLE , "Conference On Lasers And Electro Optics Europe Technical Digest") OR EXCLUDE (EXACTSRCTITLE , "ICASSP IEEE International Conference On Acoustics Speech And Signal Processing Proceedings") OR EXCLUDE (EXACTSRCTITLE , "International Conference On Thermoelectrics ICT Proceedings") OR EXCLUDE (EXACTSRCTITLE , "OSA Trends In Optics And Photonics Series")) AND (LIMIT-TO (LANGUAGE , "English")) AND (EXCLUDE (EXACTKEYWORD , "Liquid Crystals") OR EXCLUDE (EXACTKEYWORD , "Crystal Structure") OR EXCLUDE (EXACTKEYWORD , "Single Crystals") OR EXCLUDE (EXACTKEYWORD , "Liquid Crystal Displays") OR EXCLUDE (EXACTKEYWORD , "X Ray Diffraction") OR EXCLUDE (EXACTKEYWORD , "Crystals") OR EXCLUDE (EXACTKEYWORD , "Quartz") OR EXCLUDE (EXACTKEYWORD , "X Ray Photoelectron Spectroscopy") OR EXCLUDE (EXACTKEYWORD , "Silicon") OR EXCLUDE (EXACTKEYWORD , "Scanning Electron Microscopy") OR EXCLUDE (EXACTKEYWORD , "Thin Films") OR EXCLUDE (EXACTKEYWORD , "Photonic Crystals") OR EXCLUDE (EXACTKEYWORD , "Silicon Wafers") OR EXCLUDE (EXACTKEYWORD , "Crystallization") OR EXCLUDE (EXACTKEYWORD , "Heat Transfer") OR EXCLUDE (EXACTKEYWORD , "Substrates") OR EXCLUDE (EXACTKEYWORD , "Antennas") OR EXCLUDE (EXACTKEYWORD , "Wireless Telecommunication Systems") OR EXCLUDE (EXACTKEYWORD , "Reynolds Number") OR EXCLUDE (EXACTKEYWORD , "Cooling") OR EXCLUDE (EXACTKEYWORD , "Crystal Oscillators")

Continuation of Table A.1.: Search terms of main search

DB	Search term
	OR EXCLUDE (EXACTKEYWORD , "Signal Processing") OR EXCLUDE (EXACTKEYWORD , "Natural Frequencies") OR EXCLUDE (EXACTKEYWORD , "Crystal") OR EXCLUDE (EXACTKEYWORD , "Crystal Orientation") OR EXCLUDE (EXACTKEYWORD , "Chemistry") OR EXCLUDE (EXACTKEYWORD , "Quartz Crystal Microbalances") OR EXCLUDE (EXACTKEYWORD , "Temperature") OR EXCLUDE (EXACTKEYWORD , "Crystal Growth") OR EXCLUDE (EXACTKEYWORD , "Unclassified Drug") OR EXCLUDE (EXACTKEYWORD , "Atomic Force Microscopy") OR EXCLUDE (EXACTKEYWORD , "Crystallography") OR EXCLUDE (EXACTKEYWORD , "Models, Molecular") OR EXCLUDE (EXACTKEYWORD , "Refractive Index") OR EXCLUDE (EXACTKEYWORD , "X Ray Diffraction Analysis") OR EXCLUDE (EXACTKEYWORD , "Light Sources") OR EXCLUDE (EXACTKEYWORD , "Heat Transfer Coefficients") OR EXCLUDE (EXACTKEYWORD , "Gas Turbines") OR EXCLUDE (EXACTKEYWORD , "Transmission Electron Microscopy") OR EXCLUDE (EXACTKEYWORD , "Microstructure") OR EXCLUDE (EXACTKEYWORD , "Optical Properties") OR EXCLUDE (EXACTKEYWORD , "Crystallography, X-Ray") OR EXCLUDE (EXACTKEYWORD , "Animals") OR EXCLUDE (EXACTKEYWORD , "Electric Conductivity") OR EXCLUDE (EXACTKEYWORD , "Electric Fields") OR EXCLUDE (EXACTKEYWORD , "Single Crystal Silicon") OR EXCLUDE (EXACTKEYWORD , "Nanostructured Materials") OR EXCLUDE (EXACTKEYWORD , "Nanotechnology") OR EXCLUDE (EXACTKEYWORD , "Microwaves") OR EXCLUDE (EXACTKEYWORD , "Microelectromechanical Devices") OR EXCLUDE (EXACTKEYWORD , "Nematic Liquid Crystals") OR EXCLUDE (EXACTKEYWORD , "Metabolism") OR EXCLUDE (EXACTKEYWORD , "Powders") OR EXCLUDE (EXACTKEYWORD , "Liquid Crystal Polymers") OR EXCLUDE (EXACTKEYWORD , "Molecular Dynamics") OR EXCLUDE (EXACTKEYWORD , "Crystal Resonators") OR EXCLUDE (EXACTKEYWORD , "Turbomachine Blades") OR EXCLUDE (EXACTKEYWORD , "Thermochromic Liquid Crystals") OR EXCLUDE (EXACTKEYWORD , "Piezoelectricity") OR EXCLUDE (EXACTKEYWORD , "Antenna Arrays") OR EXCLUDE (EXACTKEYWORD , "Mechanical Properties") OR EXCLUDE (EXACTKEYWORD , "Morphology") OR EXCLUDE (EXACTKEYWORD , "Oxide Minerals") OR EXCLUDE (EXACTKEYWORD , "Photonic Crystal Fibers") OR EXCLUDE (EXACTKEYWORD , "Energy Gap") OR EXCLUDE (EXACTKEYWORD , "Copper") OR EXCLUDE (EXACTKEYWORD , "Spectroscopy") OR EXCLUDE (EXACTKEYWORD , "Millimeter Waves") OR EXCLUDE (EXACTKEYWORD , "X Ray Crystallography") OR EXCLUDE (EXACTKEYWORD , "CMOS Integrated Circuits") OR EXCLUDE (EXACTKEYWORD , "Crystalline Materials") OR EXCLUDE (EXACTKEYWORD , "Radar") OR EXCLUDE (EXACTKEYWORD , "Microwave Antennas") OR EXCLUDE (EXACTKEYWORD , "Dielectric Materials") OR EXCLUDE (EXACTKEYWORD , "Zinc Oxide") OR EXCLUDE (EXACTKEYWORD , "Surface Roughness") OR EXCLUDE (EXACTKEYWORD , "Optical Fibers")

Continuation of Table A.1.: Search terms of main search

DB	Search term
	OR EXCLUDE (EXACTKEYWORD , "Oxygen") OR EXCLUDE (EXACTKEYWORD , "Microstrip Antennas") OR EXCLUDE (EXACTKEYWORD , "MIMO Systems") OR EXCLUDE (EXACTKEYWORD , "Air Conditioning") OR EXCLUDE (EXACTKEYWORD , "Frequency Division Multiplexing") OR EXCLUDE (EXACTKEYWORD , "Orthogonal Frequency Division Multiplexing") OR EXCLUDE (EXACTKEYWORD , "Signal To Noise Ratio") OR EXCLUDE (EXACTKEYWORD , "Time Division Multiplexing") OR EXCLUDE (EXACTKEYWORD , "Frequency Domain Analysis") OR EXCLUDE (EXACTKEYWORD , "Hardware") OR EXCLUDE (EXACTKEYWORD , "Photons") OR EXCLUDE (EXACTKEYWORD , "Energy Utilization") OR EXCLUDE (EXACTKEYWORD , "Cameras")) AND (EXCLUDE (EXACTSRCTITLE , "Proceedings Of The IEEE Particle Accelerator Conference") OR EXCLUDE (EXACTSRCTITLE , "Advanced Materials Interfaces") OR EXCLUDE (EXACTSRCTITLE , "Nature") OR EXCLUDE (EXACTSRCTITLE , "Metrologia") OR EXCLUDE (EXACTSRCTITLE , "Digest Of Papers IEEE Radio Frequency Integrated Circuits Symposium") OR EXCLUDE (EXACTSRCTITLE , "International Archives Of The Photogrammetry Remote Sensing And Spatial Information Sciences ISPRS Archives") OR EXCLUDE (EXACTSRCTITLE , "Proceedings Of The IEEE International Frequency Control Symposium And Exposition") OR EXCLUDE (EXACTSRCTITLE , "2011 Conference On Lasers And Electro Optics Europe And 12th European Quantum Electronics Conference CLEO Europe Eqec 2011") OR EXCLUDE (EXACTSRCTITLE , "IEEE Transactions On Dielectrics And Electrical Insulation") OR EXCLUDE (EXACTSRCTITLE , "Telecommunications And Radio Engineering English Translation Of Elektrosvyaz And Radiotekhnika") OR EXCLUDE (EXACTSRCTITLE , "IEEE Antennas And Propagation Society AP S International Symposium Digest") OR EXCLUDE (EXACTSRCTITLE , "Ieej Transactions On Sensors And Micromachines") OR EXCLUDE (EXACTSRCTITLE , "International Journal Of Refrigeration") OR EXCLUDE (EXACTSRCTITLE , "American Society Of Mechanical Engineers Pressure Vessels And Piping Division Publication PVP") OR EXCLUDE (EXACTSRCTITLE , "Irmmw Thz 2010 35th International Conference On Infrared Millimeter And Terahertz Waves Conference Guide")) AND (EXCLUDE (EXACTSRCTITLE , "Ecs Transactions"))

Continuation of Table A.1.: Search terms of main search

DB	Search term
Web of Science	(TI=((((("agile" OR "agility" OR "extreme programming" OR "XP" OR "feature driven development" OR "FDD" OR "scrum" OR "crystal" OR "pair programming" OR "test-driven development" OR "TDD" OR "leanness" OR "lean software development" OR "lean development" OR "LSD") NOT "manufacturing") OR ((("agile" OR "agility" OR "extreme programming" OR "XP" OR "feature driven development" OR "FDD" OR "scrum" OR "crystal" OR "pair programming" OR "test-driven development" OR "TDD" OR "leanness" OR "lean software development" OR "lean development" OR "LSD") NOT "manufacturing") AND ("large scale" OR "scaling")) OR ("Crystal Family" OR "Dynamic Systems Development Method Agile Project Framework for Scrum" OR "Scrum of Scrums" OR "Enterprise Scrum" OR "Agile Software Solution Framework" OR "Large-Scale Scrum" OR "Scaled Agile Framework" OR "Disciplined Agile" OR "Spotify Model" OR "Mega Framework" OR "Enterprise Agile Delivery and Agile Governance Practice" OR "Recipes for Agile Governance in the Enterprise" OR "Continuous Agile Framework" OR "Scrum at Scale" OR "Enterprise Transition Framework" OR "ScALeD Agile Lean Development" OR "eXponential Simple Continuous Autonomous Learning Ecosystem" OR "Lean Enterprise Agile Framework" OR "Nexus" OR "FAST Agile")) AND ((measur* OR metric* OR "diagnostic" OR monitor* OR "quantitative" OR indicator* OR "performance indicator" OR "KPI") OR (aim OR target OR object* OR goal))))) OR AB=((((("agile" OR "agility" OR "extreme programming" OR "XP" OR "feature driven development" OR "FDD" OR "scrum" OR "crystal" OR "pair programming" OR "test-driven development" OR "TDD" OR "leanness" OR "lean software development" OR "lean development" OR "LSD") NOT "manufacturing") OR ((("agile" OR "agility" OR "extreme programming" OR "XP" OR "feature driven development" OR "FDD" OR "scrum" OR "crystal" OR "pair programming" OR "test-driven development" OR "TDD" OR "leanness" OR "lean software development" OR "lean development" OR "LSD") NOT "manufacturing") AND ("large scale" OR "scaling")) OR ("Crystal Family" OR "Dynamic Systems Development Method Agile Project Framework for Scrum" OR "Scrum of Scrums" OR "Enterprise Scrum" OR "Agile Software Solution Framework" OR "Large-Scale Scrum" OR "Scaled Agile Framework" OR "Disciplined Agile" OR "Spotify Model" OR "Mega Framework" OR "Enterprise Agile Delivery and Agile Governance Practice" OR "Recipes for Agile Governance in the Enterprise" OR "Continuous Agile Framework" OR "Scrum at Scale" OR "Enterprise Transition Framework" OR "ScALeD Agile Lean Development" OR "eXponential Simple Continuous Autonomous Learning Ecosystem" OR "Lean Enterprise Agile Framework" OR "Nexus" OR "FAST Agile")) AND ((measur* OR metric* OR "diagnostic" OR monitor* OR "quantitative" OR indicator* OR "performance indicator" OR "KPI"))))

Continuation of Table A.1.: Search terms of main search

DB	Search term
IEEE	(((((("Document Title":agile OR "Abstract":agile OR "Document Title":agility OR "Abstract":agility OR "Document Title":"extreme programming" OR "Abstract":"extreme programming" OR "Document Title":XP OR "Abstract":XP OR "Document Title":"feature driven development" OR "Abstract":"feature driven development" OR "Document Title":FDD OR "Abstract":FDD OR "Document Title":scrum OR "Abstract":scrum OR "Document Title":crystal OR "Abstract":crystal OR "Document Title":"pair programming" OR "Abstract":"pair programming" OR "Document Title":"test-driven development" OR "Abstract":"test-driven development" OR "Document Title":TDD OR "Abstract":TDD OR "Document Title":leanness OR "Abstract":leanness OR "Document Title":"lean software development" OR "Abstract":"lean software development" OR "Document Title":"lean development" OR "Abstract":"lean development" OR "Document Title":LSD OR "Abstract":LSD) NOT manufacturing) AND ("large scale" OR scaling)) AND ("Document Title":metric* OR "Abstract":metric* OR "Document Title":diagnostic OR "Abstract":diagnostic OR "Document Title":quantitative* OR "Abstract":quantitative* OR "Document Title":indicator* OR "Abstract":indicator* OR "Document Title":"performance indicator" OR "Abstract":"performance indicator" OR "Document Title":"reporting" OR "Abstract":"reporting" OR "Document Title":KPI OR "Abstract":KPI OR "Document Title":target OR "Author Keywords":target OR "Document Title":objective OR "Author Keywords":objective OR "Document Title":goal OR "Author Keywords":goal))))) OR ((((((("Document Title":agile OR "Abstract":agile OR "Document Title":agility OR "Abstract":agility OR "Document Title":"extreme programming" OR "Abstract":"extreme programming" OR "Document Title":XP OR "Abstract":XP OR "Document Title":"feature driven development" OR "Abstract":"feature driven development" OR "Document Title":FDD OR "Abstract":FDD OR "Document Title":scrum OR "Abstract":scrum OR "Document Title":crystal OR "Abstract":crystal OR "Document Title":"pair programming" OR "Abstract":"pair programming" OR "Document Title":"test-driven development" OR "Abstract":"test-driven development" OR "Document Title":TDD OR "Abstract":TDD OR "Document Title":leanness OR "Abstract":leanness OR "Document Title":"lean software development" OR "Abstract":"lean software development" OR "Document Title":"lean development" OR "Abstract":"lean development" OR "Document Title":LSD OR "Abstract":LSD) NOT manufacturing)) AND ("Document Title":metric* OR "Abstract":metric* OR "Document Title":diagnostic OR "Abstract":diagnostic OR "Document Title":quantitative* OR "Abstract":quantitative* OR "Document Title":indicator* OR "Abstract":indicator* OR "Document Title":"performance indicator" OR "Abstract":"performance indicator" OR "Document Title":"reporting" OR "Abstract":"reporting" OR "Document Title":KPI OR "Abstract":KPI OR "Document Title":target OR "Author Keywords":target OR "Document Title":objective OR "Author Keywords":objective OR "Document Title":goal OR "Author Keywords":goal))))

Continuation of Table A.1.: Search terms of main search

DB	Search term
	OR ((("Crystal Family" OR "Dynamic Systems Development Method Agile Project Framework for Scrum" OR "Scrum of Scrums" OR "Enterprise Scrum" OR "Agile Software Solution Framework" OR "Large-Scale Scrum" OR "Scaled Agile Framework" OR "Disciplined Agile" OR "Spotify Model" OR "Mega Framework" OR "Enterprise Agile Delivery and Agile Governance Practice" OR "Recipes for Agile Governance in the Enterprise" OR "Continuous Agile Framework" OR "Scrum at Scale" OR "Enterprise Transition Framework" OR "ScALeD Agile Lean Development" OR "eXponential Simple Continuous Autonomous Learning Ecosystem" OR "Lean Enterprise Agile Framework" OR "Nexus" OR "FAST Agile") AND ("Document Title":metric* OR "Abstract":metric* OR "Document Title":diagnostic OR "Abstract":diagnostic OR "Document Title":quantitative* OR "Abstract":quantitative* OR "Document Title":indicator* OR "Abstract":indicator* OR "Document Title":"performance indicator" OR "Abstract":"performance indicator" OR "Document Title":"reporting" OR "Abstract":"reporting" OR "Document Title":KPI OR "Abstract":KPI OR "Document Title":target OR "Author Keywords":target OR "Document Title":objective OR "Author Keywords":objective OR "Document Title":goal OR "Author Keywords":goal)))

Table A.1.: Search terms of main search

A.2. Context of primary sources

Search terms of the main search

ID	Research Method	Company name	Business area	Organization size	Number of teams	Team size	Agile method	Time of transformation	LSA
[S26]	Case study	Israeli Airforce	Military	60	1	15	Extreme programming	2004	No
[S12]	Case Study		Telecommunication	350			Scrum and SoS, Agile portfolio management		Yes
			Electronics	4500			Scrum, SAFe, Agile portfolio management		Yes
			Telecommunication	26000			Scrum, MoP, Agile portfolio management		Yes
			Financial	650			Scrum, Agile portfolio management		Yes
			Government	30000			Scrum, SAFe, Agile portfolio management		Yes
			Aviation	2000			Scrum, Custom, Agile portfolio management		Yes
			IT services	13000			Scrum, Spotify, Agile portfolio management		Yes

Continuation of Table A.2.: Primary studies organizational background

ID	Research Method	Company name	Business area	Organization size	Number of teams	Team size	Agile method	Time of transformation	LSA
			Transportation	30000			Kanban, MoP, Agile portfolio management		Yes
			Logistics	11000			Scrum, PMI, Agile portfolio management		Yes
			E-commerce	1100			Scrum, Spotify, Agile portfolio management		Yes
[S16]	Experience report	Hewlett Packard	Software development	3000	150		Scrum		Yes
[S19]	Case Study		Secure communication	600	1	6-7	Scrum	2008	No
			Software development	900	1	9	Ad-hoc process following agile principles		No
			Software development		1	10	ScrumBan and ScrumBut		No
			Software development	100000	8		Scrum		Yes
[S2]	Experience report	Citrix online	Software development		44		Enterprise scrum	2007	Yes
[S62]	Case study			300	5	9	Scrum, Cycles of control	2003	Yes
[S77]	Experience report	Systematic	Software development				Scrum	2005	No

Continuation of Table A.2.: Primary studies organizational background

ID	Research Method	Company name	Business area	Organization size	Number of teams	Team size	Agile method	Time of transformation	LSA
[S27]	Case study	ÜBİTAK-BİLGEM-YTE	Research and development	150	1	6	Scrum		No
[S78]	Case study		IT service provider		1	10	Extreme programming		No
			Manufacturing		1	7-11	Scrum		No
[S11]	Experience report	Capital one auto finance	Auto lender	500			Scrum, Agile portfolio management	2005	Yes
[S79]	Experience report		Power grid operator				Extreme programming		No
[S41]	Case study		Consulting		1		Scrum		No
			Web and mobile		1	5	Scrum and Kanban		No
[S80]	Case study		Software development		1	6-7	Scrum		No
			Software development		1	9	Customized agile		No
[S63]	Experience report	VeriSign	IT Security	500			Scrum	2006	Yes
[S48]	Experience report	Adobe Systems	Software development		26		Scrum	2005	Yes
[S44]	Case study		multiple		6-52	2-7	Scrum		No

Continuation of Table A.2.: Primary studies organizational background

ID	Research Method	Company name	Business area	Organization size	Number of teams	Team size	Agile method	Time of transformation	LSA
[S13]	Case study	Achmea	Financial				Scrum, Agile portfolio management	2015/2016	Yes
		AEGON	Financial				Scrum, Kanban, Lean, Agile portfolio management	2012/2013	Yes
[S81]	Case study		Cruise line industry				Scrum		No
[S4]	Case study	Vodafone	Telecommunication				SAFe		Yes
[S58]	Experience report	Microsoft	Software development				Agile	2004	No
[S37]	Case study		IT services				Agile		No
[S33]	Experience report	IONA Technology	Middleware product development				Extreme programming	1999/2000	No
[S82]	Case study		Banking		2	10	Extreme programming, TDD		No
[S73]	Survey		multiple				Scrum, agile		No
[S83]	Case study	Nokia Siemens Networks	Telecommunication		10	6-10	Scrum, TDD		Yes
[S32]	Case study		Software development		1	5	TDD		No

Continuation of Table A.2.: Primary studies organizational background

ID	Research Method	Company name	Business area	Organization size	Number of teams	Team size	Agile method	Time of transformation	LSA
[S84]	Case study		Banking, mechanical engineering, electrical engineering, digital agency	250 - 49000			Scrum, SAFe, LeSS	post 2013	Yes
[S66]	Case study	Cisco's Voice Technology		15000			Hybrid: Agile and waterfall		Yes
[S30]	Case study		Information systems development		1	8 - 10	Scrum		No
[S85]	Experience report	Escrow.com	Web development				TDD	1999 - 2002	No
			Government				TDD	2002	No
[S64]	Experience report	WMS Gaming	Game development	18	2	6 - 9	Kanban	2009	No
[S49]	Survey						Agile		
[S75]	Case study	TUBITAK SAGE software development	Software development				Scrum		
[S54]	Mixed method design		Banking				Scrum		
[S86]	Case study	Siemens					Agile		Yes

Continuation of Table A.2.: Primary studies organizational background

ID	Research Method	Company name	Business area	Organization size	Number of teams	Team size	Agile method	Time of transformation	LSA
[S24]	Case study		Banking		7		SAFe	2015	Yes
[S47]	Experience report	Cyrus Innovation LLC	IT consulting			3 - 6	Extreme programming	2003	No
[S71]	Case study		Telecommunication	50 - 150			Scrum	2004	Yes
			Telecommunication	> 150			Agile and waterfall	2008	No
	Case study		Telecommunication	>150			Scrum	2006	Yes
[S55]	Survey						Agile		No
[S51]	Experience report	ABB	Energy and automation	130000			Scrum		No
[S60]	Experience report	ThoughtWorks	Software organization			50	Extreme programming	2000	No
[S5]	Case study		IT consulting	800			Scrum, Extreme programming		No
[S53]	Survey	Multiple	Software development				Agile		No
[S39]	Survey	Multiple	Telecom, Automotive, Defence, IT-Consulting, Consulting				Agile		No

Continuation of Table A.2.: Primary studies organizational background

ID	Research Method	Company name	Business area	Organization size	Number of teams	Team size	Agile method	Time of transformation	LSA
[S20]	Case study	Ericsson AB	Telecommunication	200		80	Agile and Lean (Streamline development)	Yes	
[S22]	Case study		Software development	3000	2	60	Scrum		Yes
[S3]	Experience report	Petrobras	Oil and gas		25		Scrum	2009	Yes
[S25]	Mixed method design		Consulting	150			Scrum		Yes
[S65]	Case study		Financial		6		Spotify		Yes
[S61]	Case study		Telecommunication, Software, Transportation				Agile		Yes
[S87]	Survey						Agile		No
[S18]	Case study	Ericsson AB	Telecommunication	500		6-7	Scrum, XP	2005	Yes
[S36]	Case study	Atlassin	Software development						No
[S31]	Case study	T-Systems International	Automotive				XP, continuous integration		No
[S46]	Survey	multiple					Agile		No

Continuation of Table A.2.: Primary studies organizational background

ID	Research Method	Company name	Business area	Organization size	Number of teams	Team size	Agile method	Time of transformation	LSA
[S88]	Experience report	Siemens Healthcare GmbH	Healthcare				TDD		No
[S29]	Experience report	Quinary XP Centre	Software development		1		XP		No
[S89]	Case study	Avaya Communications Inc	Telecommunication				Agile		No
[S34]	Case study		Software development		1	10	Scrum		No
[S90]	Survey		multiple				Agile		No
[S91]	Experience report		Financial	63	9		Scrum		Yes
[S40]	Experience report		BI, Telecommunication, Government			6-8	Agile	2010	No
[S21]	Case study	Software Innovation (SI)	Software development	100	10		Scrum to kanban	2007	Yes
[S23]	Case study	Siemens			3	7-8	Scrum	2012	Yes
[S42]	Case study	multiple		50-1000			Agile or Lean		No
[S14]	Experience report	Intergraph	Software development	500	75		Scrum	2007	Yes
[S92]	Case study	Munich Re	Financial	6500	1		Agile	2007	No
[S93]	Survey	multiple					Agile		No

Continuation of Table A.2.: Primary studies organizational background

ID	Research Method	Company name	Business area	Organization size	Number of teams	Team size	Agile method	Time of transformation	LSA
[S94]	Case study		Control systems, telecommunication, wireless connectivity	3-5000			Agile, Lean		No
[S52]	Case study		IT services				Scrum		Yes
[S95]	Case study		Software development				Lean	2012	No
[S96]	Experience report	Ericsson R and D Center	Telecommunication		multiple		Kanban	2009	Yes
[S97]	Case study		E-commerce		1	6-8	Scrum		No
[S38]	Case study		Automotive	100			SAFe	2017	Yes
[S98]	Case study	Qualcomm Wireless Business Solutions			1		XP	2004	No
[S99]	Case study	Ultrasist	Software development	400	9		Agile		Yes
[S72]	Case study	Israeli Airforce	Military	60	1	15	XP	2004	No
[S67]	Experience report	Multiple					XP, Scrum		Yes

Continuation of Table A.2.: Primary studies organizational background

ID	Research Method	Company name	Business area	Organization size	Number of teams	Team size	Agile method	Time of transformation	LSA
[S100]	Experience report		Agriculture equipment manufacturing		1	35	XP	2003	No
[S35]	Mixed method design	Ericsson AB	Telecommunication	few hundred	9		Agile		Yes
[S101]	Experience report	T-mobile International	Telecommunication				TDD	2007	No
[S102]	Experience report	Davis Systems	Software development				Scrum		No
[S103]	Survey	multiple					Agile		No
[S68]	Case study	PSC	Software development	70-325	5-9	2-10	Scrum	2005-2008	Yes
[S45]	Survey						Scrum, lean		No
[S59]	Case study		Information systems development				Scrum	2013	No
[S8]	Experience report	Atlassin					Agile		No
[S56]	Case study				1	9	Scrum		No
[S50]	Case study						Agile, lean		No
S90	Experience report	Grupo Aurea	Transportation				Excecutive scrum		Yes

Continuation of Table A.2.: Primary studies organizational background

ID	Research Method	Company name	Business area	Organization size	Number of teams	Team size	Agile method	Time of transformation	LSA
		AdaptWorks	Consulting				Excecutive scrum		Yes
[S104]	Experience report	Healthwise	Healthcare	215			Scrum, lean	2003	Yes
[S105]	Case study		IT services		1		XP, Scrum	2002	No
[S6]	Case study		Multiple		2-400		Agile portfolio management		Yes
			Retail	50000-100000	50		Agile portfolio management		Yes
[S106]	Experience report	NYSE Technologies	Financial				Agile portfolio management	2008	Yes
[S43]	Survey	Multiple					Agile		No
[S7]	Experience report	Cisco Voice Technology		2500			Kanban	2010	Yes
[S15]	Experience report	Fannie Mae	Financial				Agile, DevOps	2015	Yes
[S107]	Survey	Multiple					Scrum		No
[S28]	Case study	Israeli Airforce	Military	60	1	15	XP	2004	No
[S76]	Case study	Radiante Corporation	Telecommunication		1		XP	No	
[S108]	Experience report		Software development			50	Agile	2015	No
[S17]	Experience report	SirsiDynix, StarSoft	Software development		2	25	Yes		

Continuation of Table A.2.: Primary studies organizational background

ID	Research Method	Company name	Business area	Organization size	Number of teams	Team size	Agile method	Time of transformation	LSA
[S109]	Experience report	Primavera Systems	Enterprise management solution provider	90			Scrum	2003	Yes
[S70]	Action research		Healthcare						No
[S1]	Experience report	Citrix		204	29		Enterprise Scrum	2009	Yes
[S74]	Case study	Marelli Automotive Lighting Reutlingen	Automotive				Scrum	2017	No
[S110]	Experience report								Yes
[S69]	Experience report	Nokia	Telecommunication				SAFe		Yes
[S9]	Case study		Banking	1000			SAFe	2018	Yes
[S10]	Case study	Ericsson	Telecommunication	80-200			Agile, Lean, Streamline development		Yes

Table A.2.: Primary studies organizational background

A.3. Organizational level measured

Distribution of metrics across organizational levels

Measurement level	Metrics
Enterprise	Enterprise Story Points (1), Velocity (1), Operations velocity (1), cycle time (1), build status monitor (1), efficiency regarding cost (1)
Portfolio	Return on investment (ROI) (4), Net Promoter Score (NPS) (4), application downtime (2), customer satisfaction (2), customer Experience Index (1), velocity (1), stakeholder happiness (1), business value delivered (1), investment allocation report (1), results coming period (1), portfolio wall (1), cycle time (1), results past period (1), defect count (1), sales (1), dependency visualization (1), structural quality (1), estimation accuracy (1), # of active users (1), generated income (1)

Continuation of Table A.3.: Distribution of metrics across organizational levels

Measurement level	Metrics
Program	Defect count (18), # of tests (8), code coverage (8), estimation accuracy (7), lines of code (LOC) (5), lead time (5), # of closed defects (5), technical Debt (5), velocity (5), # of completed tasks (4), churn (4), project highlights (3), throughput (3), task board (3), # of passed tests (3), test review yields (3), integration status (2), dependency visualization (2), sprint burndown chart (2), # of tests planned (2), Execution time per test case (2), % of estimations under the threshold of accuracy (2),LOC for refactoring (2), amount of code documentation (2), defect distribution (2), build time (2), # of defects found by customer (2), burndown chart (2), individual effectiveness (2),burnup chart (2), # of failed tests (2), complexity (2), ratings and satisfaction scores (2), customer satisfaction (2), story points (2), # of Contributor/Developer (2), time to fix defects (2), % of files without critical/blocker quality rules (1), overall effort estimated (1), # of features integrated (1), % of passed tests (1), results coming period (1), business value delivered (1), bug priority distribution (1), # of memory leaks (1), mean time to failure (1), code commits/broken builds (1), build throughput (1), # of iterations in the code review phase for a commit (1), scope change report (1), % of specifications delivered on time (1), task queue time (1), cost/profit analysis (1), build interval (1), cumulative Flow Diagram (1), build performance (1), customer experience index (1), milestone success (1), actual hours of team on ideal capacity (1), project dashboard (1), customer service request (CSR) handling performance indicator (1), requirement coverage over time (1), cycle time (1), return on investment (ROI) (1), % of automated tests (1), # of Unit test errors (1), defect density (1), # of tasks planned but not integrated (1), application downtime (1), burndown ratio (1), defect size (1), integration speed (1), deferred defects (1), build lead time (1), delta cost (DC) (1), list of contributors (1), average throughput time in a comment from reviewers (1), maintenance effort (1), test automation ratio (1), mean time to restore service (1), test rig availability (1), Net Promoter Score (NPS) (1), # of tasks added during the sprint (1), overdue percentage (1), # of builds (1), # of tasks in-progress at the end of the sprint (1), turnover (1), release burndown chart (1), effort to take technical debt to zero (1), requirements waste and change requests (1),

Continuation of Table A.3.: Distribution of metrics across organizational levels

Measurement level	Metrics
	epic progress report (1), results past period (1), epic status (1), sales (1), binary file size (1), self-assessments of continuous improvement (1), estimation per story point (1), status of release steps per sprint (1), binary status of tool availability (1), system response time (1), feedback time (1), task completion time (1), files per user story (1), # of function points implemented (1), fix time of failed builds (1), generated income (1), user stories delivered (1), testing stability of a project (1), virtual actual hour cost (VAHC) (1), time between the designer's readiness of model and model's release (1), volatility Index (1), top contributors based on LOC (1), # of broken builds (1), # of GDI leaks (1), effort per use case point (1), drag Factor trend (1), virtual planned hour cost (VPHC) (1), duplicated code (1), waste (1), duration of impediments (1), earned business value (1)

Continuation of Table A.3.: Distribution of metrics across organizational levels

Measurement level	Metrics
Team	Velocity (41), defect count (38), estimation accuracy (21), complexity (21), code coverage (19), time to fix defects (18), burndown chart (16), lines of code (LOC) (12), deferred defects (11), # of tests (11), cycle time (11), defect density (11), Net Promoter Score (NPS) (8), task distribution 8, # of completed tasks (8), # of closed defects (7), task completion time (6), team satisfaction (6), availiable capacity (6), cost (6) # of granted patents (6), Cost (6), Business value delivered (5), Return on investment (ROI) (5), Amount of code documentation (5), Defect distribution (5), # of passed tests (5), Customer satisfaction (5), # of failed tests (4), # of reopened defects (4), Story points (4), Duplicated code (4), Lead time (4), Work in progress (3), Technical Debt (3), % of passed tests (3), Burnup chart (3), # of patents applications (3), # of tasks added during the sprint (3), # of collected ideas from key stakeholders (3), Cumulative Flow Diagram (3), Defect identification performance (3), # of blocking WIP (2), Functional capacity evaluation (2), # of generated ideas by the team (2), % of features delivered on time (2), Pulse metric (2), % of finished scope (2), Definition of done (2), % of impediments (2), # of completed feature specifications (2), % of specifications delivered on time (2), # of tasks in-progress at the end of the sprint (2), % of tasks completed (2), Remaining effort (2), Bucket- and fix-it list (2), Time for unforseen complexity (2), # of iterations in the code review phase for a commit (2), Downtime ratio (2), # of builds (2), Early Signal Testing Scorecard (ESTS) (2), Consumer feedback (2), Function points (2), Contribution chart (2), Instrumented Scorecard (2), # of defects older than a treshold (2), # of completed function points / story points (2), Cost adherence (2), Release burndown chart (2), "Cost Performance Index (CPI)" (2), System response time (2), # of ready tasks (2), # of Unit test errors (2), # of defects found by customer (2), User performance (2), # of commits (2), Velocity of elaboration of features (2), Commit Response Performance (1), Schedule performance index (SPI) (1), Non-issue component commits (1), Commit review speed (1), Code reliability (1), Commit size (1), # of broken builds (1), Commitment Index (1), # of meetings (1), # of changes in the development enrivonment (tools) (1), Targeted value increase (1), Consistency of Relative Estimation (1), Time between events (1), % of releases delivered on time (1), Build performance (1), % of errors identified after sprint review (1), Mean time to failure (1), Control chart (1), Performance in Lines of code (1), Core component commits (1), Resolved Issues' Throughput (1), Correlation between Story Point and Actual Effort (1), Sprint readiness (1), % of files without critical/blocker quality rules (1), Task status (1), Actual effort spend on product (1), Test review yiels (1), Cost of unplanned changes (1),

Continuation of Table A.3.: Distribution of metrics across organizational levels

Measurement level	Metrics
	Time spent for change based on client feedback (1), # of estimated tasks (1), Ideal hours / Story points (1), # of fast tests (1), Landing zone story (1), Customer involvement (1), LOC/File (1), Amount of rework (1), # of tasks reopened (1), Customer value (1), Open defect severity index (1), # of released defects (1), % of fast test builds (1), Cycle-time (1), Requirement clarity index (1), Daily build pass through time (1), # of tasks tested (1), Defect classification (1), Spend per AFP (funciton point) (1), # of releases delivered on time (1), # of tasks that are pending tests (1), Defect count during design review (1), Code maintainability (1), Defect count/Burnup (1), Task-type in a timeframe (1), Velocity vs Unplanned Effort Rate (1), # of violations regarding component calls (1), # of remaining tasks (1), Testing stability of a project (1), Thumbs-up Rule (1), Time needed for deployments (1), Defect density per release (1), Total effort spent in testing (1), Defect detection percentage (DDP) (1), Ideal Days (1), # of function points implemented (1), Innovation Rate (1), Amount of users (1), Integration speed (1), defect removal efficiency (1), Build status monitor (1), Defects found in peer review (1), LOC removed in refactoring (1), Application server capacity (1), Mainline breakage (1), Automation speed (1), Milestone success (1), Delivery on time (1), Non-compliance Index (1), Delivery time and deadlines to achieve value (1), Number feedbacks provided by testers to developers on the same issue (1), Dependency visualization (1), Pain index (1), Developer-testser communication cycle (1), Pomodoro (1), # of iterations in the code review phase (1), Real invested effort of an issue/story point (1), Average usage days vs User count (1), Churn (1), Backlog (1), Requirement readiness (1), Effort remaining (1), Resolved tasks throughput (1), Effort spend (1), RUF metrics (Reliability, usability and Functionality (1), # of critical defects (1), Software defect removal efficiency (1), % of automated tests (1), Sprint burndown chart (1), Employee interaction (1), Sprint report (1), End user feedback (1), # of generated ideas from/based on third party (1), Epic burndown chart (1), Task board (1), Epic progress report (1), # of tests planned (1), Backlog map (1), Tasks finished flow comparison (1), Estimation per story point (1), Team effectiveness (1), Execution time per test case (1), Team status (1), Feature and epic progress (1), Test automation ratio (1), Feature backlog (1), Test-Growth-Ratio (1), Fix time of failed builds (1), Throughput (1), Flow efficiency (1), Code security (1), Focus factor (1), Time needed for non-implementation (1), Forecast horizon (1), % of adopted work (1), Blocked issues chart (1), TQI Score (1), % of estimations under the treshold of accuracy (1), Commit Review Performance (1), Hardware (1), Effort spend for quality activities (1), Employee experience (1), Working software delivery success rate (1), Defect density after internal delivery (1), Defect density in daily build (1),

Continuation of Table A.3.: Distribution of metrics across organizational levels

Measurement level	Metrics
Individual	Available capacity (3), individual effort remaining (1), individual effectiveness (1), developer reputation (1), time to fix defects (1), employee cost (1)

Table A.3.: Distribution of metrics across organizational levels

A.4. Definition of metrics

Description of metrics

Metric	Description	Primary study
Number of defects	Number of defects per iteration	[S26] [S16] [S19] [S79] [S48] [S83] [S30] [S85] [S71] [S20] [S25] [S46] [S88] [S89] [S91] [S40] [S21] [S23] [S38] [S100] [S35] [S68] [S50] [S104] [S15] [S28] [S109] [S9]
Velocity	Number of completed features/user stories per iteration	[S26] [S12] [S19] [S2] [S27] [S78] [S11] [S79] [S41] [S13] [S58] [S33] [S82] [S73] [S64] [S75] [S47] [S60] [S20] [S25] [S65] [S87] [S89] [S42] [S14] [S52] [S97] [S98] [S101] [S68] [S104] [S106] [S43] [S76] [S108] [S17] [S1] [S69]
Estimation accuracy	Actual story points / estimated story points	[S12] [S19] [S13] [S81] [S4] [S73] [S75] [S29] [S89] [S14] [S97] [S38] [S100] [S43] [S28] [S76]
Code coverage	Number/Percentage of code covered by test	[S16] [S19] [S79] [S32] [S30] [S85] [S3] [S25] [S18] [S31] [S88] [S91] [S40] [S93] [S101] [S102] [S28]
Code complexity	Number of classes, interfaces, methods	[S19][S32] [S30] [S25] [S31] [S40] [S100] [S101] [**652**]
Burndown chart	Project remaining work vs remaining time	[S26] [S12] [S73] [S32] [S75] [S22] [S25] [S87] [S34] [S92] [S97] [S72] [S35] [S101] [S104] [S28] [S108] [S17] [S109] [S1]
Time to fix defects	time spent between when an error is identified and when it is corrected	[S19] [S79] [S41] [S33] [S73] [S83] [S71] [S87] [S46] [S40] [S38] [S100] [S68] [S50]
Number of tests	The number of test cases/test points	[S26] [S19] [S30] [S20] [S3] [S87] [S31] [S89] [S91] [S40] [S38] [S28]
Lines of Code (LOC)	Number of non-comment source code statements	[S16] [S19] [S30] [S3] [S18] [S31] [S88] [S40] [S100] [S17]
Cycle time	Time when work is ready for delivery - time when work is started	[S12] [S19] [S2] [S11] [S64] [S51] [S46] [S103] [S56] [S69] [S43] [S15] [S109]
Defect density	Amount of defects per LOC or function point	[S19] [S27] [S73] [S30] [S88] [S91] [S102] [S15]
Deferred defects	Number of defects not solved and carried over to the next iteration	[S19] [S41] [S73] [S25] [S87] [S18] [S38] [S102] [S68] [S108]
Number of completed tasks	Number of completed tasks	[S19] [S89] [S21] [S14] [S35] [S68]
Net Promoter Score (NPS)	Users have to answer the question "'How likely would you be to recommend our company/product to a friend or colleague?". According to the score (between 0 and 10) that users provide, they are classified as promoters (9-10), passive (7-8) or detractors (0-6). The overall Net Promoter Score is the percentage of promoters minus the percentage of detractors.	[S48] [S37] [S73] [S36] [S46] [S95] [S102] [S8] [S6] [S43] [S9]

Continuation of Table A.4.: Metric definition

Metric	Description	Primary study
Number of closed defects	The amount of fixed defects within a build/iteration	[S16] [S19] [S30] [S20] [S25] [S87] [S21] [S68]
Return on Investment (ROI)	Net income/investment	[S4] [S5] [S39] [S46] [S23] [S94] [S6] [S7] [S15]
Available capacity		[S12] [S73] [S75] [S68]
Customer satisfaction	Mostly measured by surveys	[S11] [S73] [S87] [S46] [S103] [S104] [S108] [S9]
Technical debt	Amount and overview of technical debt identified	[S19] [S73] [S3] [S23]
Task distribution	Number of submitted, verified, active and implemented tasks per week visualized in a chart	[S19] [S33] [S51] [S25] [S35] [S76]
Number of passed tests	The number of successfully executed tests	[S16] [S19] [S87] [S38] [S72] [S35]
Lead time	The time per task from the moment of definition to delivery or between steps	[S19] [S75] [S21] [S96] [S38] [S56] [S43] [S9]
Amount of code documentation	Number or percentage of documentation comment lines	[S19] [S32] [S38]
Task completion time	Time to task completion	[S19] [S36] [S68] [S45] [S8] [S56]
Business value delivered	Business value of all developed stories	[S13] [S87] [S46] [S106] [S43]
Defect distribution	Number of different bugs (non-functional, performance, security,...)	[S51] [S31] [S91] [S40] [S17]
Number of granted patents	The number of granted patent applications	[S94]
Team satisfaction	Satisfaction survey	[S12] [S13] [S103] [S9]
Story points	Individual size measure (e.g. estimated in team monts/ amount a developer can develop in a day, ...)	[S80] [S87] [S14] [S106] [S1]
Number of failed tests	Number of failed tests	[S19] [S3] [S89] [S68]
Cost	Operating expenses by a period, dollar spend	[S13] [S46] [S15] [S9]
Burnup chart	Time spend on non-sprint planned activities	[S75] [S14] [S38] [S35]
Duplicated code	Number of duplicated code lines (10 LOC or more)	[S19] [S40]
Churn	Number of lines or files added, changed or removed	[S16] [S19] [S46] [S21]
Test review yields	Overview of test results	[S12] [S16] [S38] [S102]
Dependency visualization	Visualized dependencies between teams and tasks	[S12] [S24] [S46]
Number of tasks added during the sprint	Number of tasks added during the sprint	[S19]
Number of reopened defects	Number/percentage of reopened defects	[S19] [S25] [S102]
Number of defects found by customer	A measure of defects found by the customer	[S38] [S102] [S104] [S109]
Task board	Visualization of all tasks for the current sprint	[S16][S34] [S96]
Cumulative flow diagram	The visualized flow of story implementation	[S77] [S75] [S96] [S69]
Percentage of passed tests	Percentage of passed tests	[S16] [S41] [S89] [S68]
Number of tests planned	Number of planned tests (function test cases)	[S20]

Continuation of Table A.4.: Metric definition

Metric	Description	Primary study
Percentage of estimation under the threshold of accuracy		[S19]
Number of collected ideas from key stakeholders		[S94]
Execution time per test case	Execution time per test case, unit test	[S19] [S38]
Release burndown chart		[S75] [S109]
Individual effectiveness	Total churn per developer per quarter, production per developer	[S21] [S68]
System response time	Response time for the server	[S19] [S25] [S38]
Work in progress	Amount of work in progress	[S19] [S75] [S46]
Number of unit test errors	Number of unit test failures	[S19]
Number of builds	Number of builds	[S19] [S15]
Number of patent applications	Number of patent applications	[S94]
Number of tasks in progress at the end of the sprint	Number of tasks in progress at the end of the sprint	[S19] [S97]
Throughput	Throughput time for an issue/ticket/feature	[S19] [S9]
Number of commits	Number of commits on core components, number of commits that are not related to an identified issue	[S19]
Mean time to failure	Time difference between the occurrence of failures	[S25] [S38]
Test automation ratio	Percentage of automated tests	[S102] [S9]
Number of iterations in the code review phase for a commit	Number of iterations in the code review phase for a commit	[S19]
Contribution chart	Contributors per file of code	[S75] [S51]
Bucket- and fix-it list	List containing sources teams need (money, time, specialists)	[S13]
Number of completed function points / story points	Number of completed function points / story points	[S45] [S15]
Build performance	Percentage of build success vs failure, or builds below the treshold	[S16] [S41]
Number of defects older than a threshold	Number of defects older than a threshold	[S19]
Build status monitor	Monitor with rectangles showing the build status in green or red	[S3] [S34]
Consumer feedback	(Number of) feedback from end user	[S19] [S46]

Continuation of Table A.4.: Metric definition

Metric	Description	Primary study
Build time	Build time, compilation time	[S19]
Results coming period	results coming period	[S12]
Downtime ratio	Total downtime related to total runtime	[S25]
Application downtime	Application downtime, product downtime per month	[S9] [S10]
Number of broken builds	Number of broken builds during continuous integration, number of crashes	[S31] [S23]
Percentage of files without critical/blocker quality rules	Percentage of files without critical/blocker quality rules	[S19]
Testing stability of a project	Testing stability of a project	[S19]
Number of blocking WIP	Number of blocking WIP	[S58]
Time for unforeseen complexity	Time for fixed complexity that could have been identified during initial scoping, time for overhead activities	[S47] [S28]
LOC for refactoring	Lines of Code for refactoring	[S16]
Percentage of specifications delivered on time	Percentage of specifications delivered on time	[S19]
Milestone success	Ratio of milestones reached as planned	[S12] [S25]
Estimation per story point	Estimated effort of an issue/story point	[S19]
Pulse metric	Number of check-ins per day into the integration branch	[S26] [S72]
Number of contributor/Developer	Number of contributor/Developer in general and per feature/theme	[S16]
Remaining effort	Number of remaining points per sprint, remaining task effort	[S35] [S68]
Fix time of failed builds	Fix time of failed builds	[S77]
Results past period	Results past period	[S12]
Function points	Function points	[S87] [S46]
User performance	User performance questionnaires	[S36] [S107]
Functional capacity evaluation	Who is working, etc...	[S13]
Cost Performance Index (CPI)	Earned value relative to actual cost	[S25] [S46]
Percentage of tasks completed	Percentage of stories completed	[S89] [S97]
Number of function points implemented	Number of function points implemented	[S11]
Instrumented scorecard	Product analytics and log file analysis of user interactions of with the product	[S36] [S8]

Continuation of Table A.4.: Metric definition

Metric	Description	Primary study
Percentage of finished scope	Percentage of the estimated scope of the product to be delivered with, planned-to-done feature ratio	[S46] [S103]
Integration speed	Days needed to integrate	[S38] [S35]
Number of generated ideas by the team	Number of generated ideas by the team	[S94]
Integration status	Generation time, rig status	[S38]
Number of completed feature specifications	Number of completed feature specifications	[S19]
Early Signal Testing Scorecard (ESTS)	Task completion percentage, UMUX-lite, Time on task, Expetation score, Happy path clicks. All divided into New and Existing users.	[S36] [S8]
Percentage of impediments	Percentage of impediments	[S12]
Epic progress report	We use epics to capture the main themes of a release. We are interested in progress (to 100%) of the user stories associated with each epic. The data labels show the total number of points for this epic in this release.	[S75] [S14]
Number of ready tasks	Number of tickets in the "ready" colums	[S19]
Velocity of elaboration of features	Velocity of elaboration of features	[S77]
Cost adherence	Adherence to budget	[S81] [S39]
Definition of done	(1) Code is implemented, working, commented, and refactored; (2) Code is complying with coding guidelines; (3) Code is reviewed by another developer (4-eye principle); (4) Module-Tests are implemented (if possible), (5) UI-test with a set of specified inputs; (6) (New) functionality is part of the test-checklist considered for regression-testing; (7) All used Excel-versions are tested.	[S79] [S32]
Structural quality	Visual heat-map of applications for quality and size. Enterprise trending of scores—a 2D line chart of overall code quality over time for all applications presented by portfolio, sub-portfolio, or application	[S15]
Operations velocity	Velocity of operations	[S1]
Core component commits	Density of core component commits in a given period before the planned release	[S41]
Percentage of features delivered on time	Percentage of features delivered on time	[S19]

Continuation of Table A.4.: Metric definition

Metric	Description	Primary study
Requirement readiness	Requirement readiness	[S75]
Cost of unplanned changes	Cost of unplanned changes	[S50]
Test-Growth-Ratio	Number of tests/Number of source code	[S31]
Number of features integrated	Number of features integrated	[S20]
Maintenance effort	Maintenance effort	[S18]
Cost/profit analysis	Cost/profit analysis by feature	[S106]
Project highlights	Project highlights	[S12]
Number of changes in the development environment	Number of changes introduced to tooling	[S38]
Scope change report	We save the scope when we complete the release planning, calling this the "baseline scope". We then report on changes to that scope, showing summary information such as a total number of points for the release, what has been added and removed. This is followed by a listing of the user stories that have been added and removed so that everyone understands the current status of the plan.	[S14]
Customer Experience Index	Customer Experience Index	[S9]
Tasks finished flow comparison	The chart contains three data sets: the blue line shows the number of tasks finished during each period, the orange line shows that given the total number of tasks finished, how many of them were stopped or blocked at some point during the period, and the black line shows how many went "clean" through the system without being stopped or blocked.	[S56]
Customer involvement	Number of times that end-user calls to support team in a given period (per month)	[S19]
Time spend for change based on client feedback	Time spend for change based on client feedback	[S47]
Number of GDI leaks	Number of GDI leaks	[S23]
Build lead time	Build lead time	[S38]
Customer service request (CSR) handling performance indicator	The Includes the key information of the CSR as well as the number of days to overdue and the reported hours. Also, the number of days the platform maintenance was involved is visualized. The overdue cases are marked with red and the soon-to-overdue with yellow.	[S96]

Continuation of Table A.4.: Metric definition

Metric	Description	Primary study
Build throughput	Build throughput	[S38]
Customer value	Points for user story	[S46]
Performance in LOC	ESLOC (Effective Source Lines Of Code)/hour	[S100]
Number of released defect	Number of defects released to the customer	[S103]
Number of iterations in the code review phase	Average number of iterations in the code review phase	[S19]
Number of violations regarding component calls	The number of illegal calls from one component to another	[S40]
Daily build pass through time	Daily build pass through time	[S19]
Sprint report	Sprint report	[S75]
Defect classification	Class of defect severity used for milestone decision-making	[S83]
Code reliability	Code reliability	[S19]
Percentage of releases delivered on time	Percentage of releases delivered on-time in a given period (e.g. year)	[S19]
Commit review speed	Commit review speed	[S19]
Defect count during design review	Design review yields	[S102]
Time between the designer's readiness of model and model's release	Time between the designer's readiness of model and model's release	[S38]
Defect count/Burnup	Number of resolved issues (internal defects) + burn-up + fluctuations in velocity	[S38]
TQI Score	Highlights the operational and cost risks of the application system in five areas: robustness, security, performance efficiency, changeability, and transferability (or comprehensibility). These five measures are aggregated into a sixth measure—the total quality index (TQI)—that provides a summary structural-quality score.	[S15]
Number of releases delivered on time	Number of releases delivered on time in a certain period	[S19]
Percentage of fast test builds	Percentage of fast test builds	[S19]
Defect density after internal delivery	Percentage of errors identified after internal delivery (minor release)	[S19]
LOC/File	LOC/File	[S32]

Continuation of Table A.4.: Metric definition

Metric	Description	Primary study
Defect density in daily build	Percentage of errors identified in daily build (developer)	[S19]
Number of tasks tested	Number of already tested tickets	[S19]
Defect density per release	Percentage of errors identified during validation for a given release	[S19]
Number of feedback provided by testers to developers on the same issue	Number feedbacks provided by testers to developers on the same issue (communication cycles between developers and testers for the same issue)	[S19]
Defect detection percentage (DDP)	Number of defects remaining / Number of defects found by testing phase	[S30]
Overdue percentage	Overdue percentage (per severity)	[S96]
Number of remaining tasks	Number of remaining tasks	[S68]
Portfolio wall	Portfolio wall	[S12]
Defect identification performance	Density of bugs that were corrected within the defined duration threshold	[S41]
Requirement clarity index	Requirement clarity index	[S73]
Defect size	Total LOC per defect	[S16]
Number of critical defects	Number of critical defects	[S68]
Defects found in peer review	Percentage of total defects found in peer review	[S102]
Sales	Sales for the service channels the users were in charge of	[S9]
Actual effort spend on product	Time spent distribution based on sprints is summed, and area chart is given since we expect to see similar level of actual effort in each sprint.	[S27]
Spend per AFP (funciton point)	Spend per AFP (funciton point)	[S15]
Virtual planned hour cost (VPHC)	Hours cost considering planned hours (number of team/delivery units to purchase from the supplier)/Planned hours	[S4]
Status of release steps per sprint	Status of release steps per sprint	[S38]
Volatility index	Volatility index	[S11]
Targeted value increase	Targeted value increase	[S73]

Continuation of Table A.4.: Metric definition

Metric	Description	Primary study
Actual hours of team on ideal capacity	Actual hours on ideal capacity Ideal team capacity in hours the supplier offers. C = S $\times(DS\check{}(NW + (DS \times KO/20))) \times TS \times DH(S = Numberofscrumteam, DS = Durationofasprintindays NW = Non - workingdaysforateam KO = Costofknowledgetransfer, planningandestimationandshowandtellsession TS = Numberofpeopleinateam)$	[S4]
Task queue time	Queue time in the backlog (per severity, average, and distribution)	[S96]
Thumbs-up Rule	Thumbs-up Rule	[S73]
Commit Response Performance	Density of commits with duration below the defined threshold	[S41]
Number of active users	Net change in active users	[S9]
Commitment index	Commitment index	[S11]
Consistency of relative estimation	Identifying the outliers, which are extreme values considering PBIs with story point and actual efforts spent on each PBI. These PBIs are determined using box plot outlier analysis.	[S27]
Amount of rework	Defined as the total effort consumed in fixing defects calculated as a percentage of total effort consumed in code writing	[S108]
Time needed for deployments	Time needed for deployments	[S47]
Drag factor trend	Effort in hours which do not contribute to sprint goal	[S11]
Top contributors based on LOC	Top contributors based on LOC	[S16]
Amount of users	Amount of users	[S95]
Control chart	Control chart	[S75]
Duration of impediments	Number of hours that do not produce a tangible outcome	[S4]
Velocity vs Unplanned Effort Rate	Team velocity versus the ratio of unplanned time spent within whole time spent in a sprint since hot fixes are currently added to current sprint.	[S27]
List of contributors	List of contributors (developer that contribute code to the build)	[S16]
Earned business value	Earned business value	[S17]
LOC removed in refactoring	Lines of code removed per one hour of refactoring	[S40]

Continuation of Table A.4.: Metric definition

Metric	Description	Primary study
Efficiency regarding cost	NPV/effort	[S1]
Mainline breakage	Mainline breakage	[S33]
Effort per use case point	UCPs = (UAW + UUCW) * TCF * ECF Where the Unadjusted Actor Weight (UAW) is the sum of the weights of the actors that are involved in the use cases, the unadjusted Use-Case Weight Total (UUCW) is the sum of the use cases' weights, the Technical Complexity Factor (TCF) is the sum of the weights of the technical fac-tors that are involved in the use cases, and the Environmental Complexity Factor (ECF) is the sum of the weights of the environmental factors that are involved in the use cases.	[S99]
Number of tasks reopened	The number of tasks that were reopened during the cycle time period. Reopens can come from other developers during code reviews or from the QA during the testing.	[S56]
Effort remaining	Team effort remaining	[S68]
Net Present Value (NPV)	Net Present Value	[S1]
Actual effort for one story point	Total number of estimated story points / actual efforts in each sprint	[S27]
Non-compliance Index	Non-compliance Index	[S73]
Effort spend for quality activities	Time allocation for verification and test activities / total efforts	[S27]
Open defect severity index	Open defect severity index	[S73]
Effort to take technical debt to zero	Effort in man days necessary to take technical debt to zero	[S3]
Overall effort estimated	The entire backlog estimate with the number of iteration/weeks to deliver the product (development + QA +UAT + production deployment).	[S106]
Employee cost	$(\text{Employee}_{s}alary \times 160)/(standard_{s}alary \times working_{h}ours)$	[S99]
Pain index	The ratio of negative NPS comments, which we refer to as the Pain Index, was used to quantify these qualitative inputs. Pain index data was then combined with additional quantitative inputs, such as product analytics, to surface the areas that require most attention.	[S36]
Employee experience	Employee experience	[S43]
Pomodoro	A pomodoro is a particular kind of story point and corresponds to 30 minutes actual work usually performed by a pair	[S29]

Continuation of Table A.4.: Metric definition

Metric	Description	Primary study
Employee interaction	Number of post in the open source forum in a given period	[S19]
Project dashboard	Project dashboard	[S67]
End user feedback	Density of end user feedback related to an issue in a given period	[S41]
Number of tasks that are pending tests	Number of tasks that are pending tests	[S19]
Enterprise Story Points	A team of 3-4 developers (including QA and userexperience staff) is 50 ESPs per month. A team of 5- 7 developers is 100 ESPs per month.	[S1]
Real invested effort of an issue/story point	Real invested effort of an issue/story point	[S19]
Epic burndown chart	Epic burndown chart	[S75]
Burndown ratio	The share of completed work against committed work	[S9]
Application server capacity	CPU capacity for performing tasks	[S25]
Requirement coverage over time	Requirements coverage over time (per sw. revision)	[S38]
Epic status	Epic status	[S12]
Requirements waste and change requests	Ratio of implemented requirements in comparison to wasted requirements. Furthermore, the change requests per requirement.	[S18]
Automation speed	Automation speed	[S19]
Number of meetings	Number of team meetings per iteration	[S25]
Number of estimated tasks	Number of issue estimation in a given period (e.g. sprint)	[S19]
RUF metrics	RUF metrics (Reliability, usability & functionality	[S36]
Average throughput time in a comment from reviewers	Average throughput time in a comment from reviewers	[S19]
Schedule performance index (SPI)	Earned value relative to planned value	[S25]
Feature and epic progress	Feature and epic progress	[S75]
Software defect removal efficiency	Percentage of defects that is removed before the software is released.	[S100]
Feature backlog	Number of estimated and remaining story points	[S35]
Sprint readiness	Percentage of stories allocated to a sprint that is ready, and is supported by a checklist	[S77]
Feedback time	Actual feedback time from CI to developers (from the beginning of test run until ist end)	[S19]

Continuation of Table A.4.: Metric definition

Metric	Description	Primary study
Stakeholder happiness	Stakeholder happiness	[S12]
Files per user story	Number of Files (NOF) per user story	[S16]
Code commits/broken builds	Code commits/broken builds	[S38]
Average usage days vs User count	Average usage days vs User count	[S36]
Percentage of adopted work	Percentage of adopted work	[S73]
Flow efficiency	Given a work item or a process, the efficiency is calculated based in the amount of time used in "working" and the amount of time in "waiting". Formula is Work Time/(Work Time + Waiting Time)	[S56]
Code maintainability	Code maintainability	[S19]
Focus factor	Focus factor	[S73]
Code security	Code security	[S19]
Forecast horizon	Forecast horizon	[S12]
Task status	Task status	[S75]
Backlos	Backlog	[S75]
Team effectiveness	Team effectiveness	[S68]
Backlog map	Backlog map	[S75]
Team status	Team status	[S75]
Generated income	Generated income per service channel	[S9]
Commit size	Commit size	[S19]
Hardware	CPO capacity for database queries	[S25]
Test rig availability	Test rig availability	[S38]
Ideal days	Ideal days	[S80]
Percentage of automated tests	Percentage of automated tests	[S43]
Ideal hours / Story points	Effort in ideal hours or story points	[S52]
Time between events	Time between events	[S75]
Binary file size	Artifact size (binary file size)	[S38]
Percentage of errors identified after the sprint review	Percentage of errors identified after the sprint review	[S19]
Individual effort remaining	Individual effort remaining	[S68]
Time needed for non-implementation	Time needed for non-implementation	[S47]
Innovation rate	Innovation rate	[S27]

Continuation of Table A.4.: Metric definition

Metric	Description	Primary study
Number of memory leaks	Number of memory leaks	[S23]
Binary status of tool availability	Binary status of tool availability	[S38]
Total effort spent in testing	total effort consumed in testing calculated / total effort consumed in code writing	[S108]
Blocked issues chart	Blocked issues chart	[S75]
Turnover	Turnover	[S9]
Number of tasks planned but not integrated	Number of features planned for the integration up to date in the integration plan (but not yet integrated)	[S20]
User stories delivered	List of user stories delivered	[S16]
Investment allocation report	The chart shows how much we are spending on the various investment allocations for the release in comparison to the planned allocation. The Actual Allocation is calculated by summing the story points for completed stories in the release in each of the categories.	[S14]
Correlation between Story Point and Actual Effort	Spearman correlation between Story Point and Actual Effort	[S27]
Landing zone story	Landing zone story	[S75]
Virtual actual hour cost (VAHC)	"Hours cost considering actual hours (Number of team/delivery units to purchase from the supplier)/Planned hours	[S4]
Bug priority distribution	Distribution of Blocker, Critical, Major, Minor, and Trivial defects	[S91]
Build interval	Build interval	[S38]
Waste	Cost originated from delta estimation precision (DEP) and Virtual actual hour cost (VAHC) (DEP VAHC)/100	[S4]
Delivery on time	Delivery on time	[S73]
Working software delivery success rate	Working software delivery success rate	[S68]
Delivery time and deadlines to achieve the VA	delivery time and deadlines to achieve the VA	[S39]
Delta cost (DC)	Cost originated from the virtual hours cost (VAHC – VPHC)/100	[S4]

Table A.4.: Metric definition

List of Figures

List of Tables

List of Abbreviations

CMMI Capability Maturity Model Integration.

EBI Enterprise Backlog Item.

GQM Goal Question Metric.

LOC Lines of Code.

NPS Net Promoter Score.

NPV Net Present Value.

PBI Product Backlog Item.

ROI Return on Investment.

SAFE Scaled Agile Framework.

SLR Scientific Literature Review.

SMS Systematic Mapping Study.

XP Extreme Programming.

Bibliography

[1] D. Cohen, M. Lindvall, and P. Costa. "An introduction to agile methods." In: *Adv. Comput.* 62.03 (2004), pp. 1–66.

[2] *10th annual "state of agile development" survey*. 2016. URL: http://www.agile247.pl/wp-content/uploads/2016/04/VersionOne-10th-Annual-State-of-Agile-Report.pdf.

[3] K. Dikert, M. Paasivaara, and C. Lassenius. "Challenges and success factors for large-scale agile transformations: A systematic literature review". In: *Journal of Systems and Software* 119 (2016), pp. 87–108.

[4] A. Meidan, J. A. Garcıa-Garcıa, I. Ramos, and M. J. Escalona. "Measuring software process: a systematic mapping study". In: *ACM Computing Surveys (CSUR)* 51.3 (2018), pp. 1–32.

[5] E. Kupiainen, M. V. Mäntylä, and J. Itkonen. "Using metrics in Agile and Lean Software Development–A systematic literature review of industrial studies". In: *Information and Software Technology* 62 (2015), pp. 143–163.

[6] M. Feyh and K. Petersen. "Lean software development measures and indicators-a systematic mapping study". In: *International Conference on Lean Enterprise Software and Systems*. Springer. 2013, pp. 32–47.

[7] Ö. Uludag, P. Philipp, A. Putta, M. Paasivaara, C. Lassenius, and F. Matthes. "Revealing the State-of-the-Art in Large-Scale Agile Development: A Systematic Mapping Study". In: *arXiv preprint arXiv:2007.05578* (2020).

[8] K. Beck, M. Beedle, A. Van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, et al. *The agile manifesto*. 2001.

[9] R. Hoda, N. Salleh, and J. Grundy. "The rise and evolution of agile software development". In: *IEEE software* 35.5 (2018), pp. 58–63.

[10] C. J. Stettina and J. Hörz. "Agile portfolio management: An empirical perspective on the practice in use". In: *International Journal of Project Management* 33.1 (2015), pp. 140–152.

[11] R. Müller, M. Martinsuo, and T. Blomquist. "Project portfolio control and portfolio management performance in different contexts". In: *Project management journal* 39.3 (2008), pp. 28–42.

[12] *14th annual "state of agile development" survey*. 2020. URL: https://stateofagile.com/#ufh-i-615706098-14th-annual-state-of-agile-report/7027494.

[13] N. Fenton and J. Bieman. *Software metrics: a rigorous and practical approach*. CRC press, 2014.

[14] S. Keele et al. *Guidelines for performing systematic literature reviews in software engineering*. Tech. rep. Citeseer, 2007.

[15] B. Kitchenham and S. Charters. "Guidelines for performing systematic literature reviews in software engineering". In: (2007).

[16] H. Zhang, M. A. Babar, and P. Tell. "Identifying relevant studies in software engineering". In: *Information and Software Technology* 53.6 (2011), pp. 625–637.

[17] J. Highsmith and A. Cockburn. "Agile software development: the business of innovation". In: *Computer* 34.9 (2001), pp. 120–127. DOI: 10.1109/2.947100.

[18] A. Cockburn. *Agile software development: the cooperative game*. Pearson Education, 2006.

[19] L. Williams and A. Cockburn. "Agile software development: it's about feedback and change". In: *IEEE computer* 36.6 (2003), pp. 39–43.

[20] K. Beck. "Embracing change with extreme programming". In: *Computer* 32.10 (1999), pp. 70–77.

[21] K. Beck. *Extreme programming explained: embrace change*. addison-wesley professional, 2000.

[22] H. Takeuchi and I. Nonaka. "The new new product development game". In: *Harvard business review* 64.1 (1986), pp. 137–146.

[23] K. Schwaber and M. Beedle. *Agile software development with Scrum*. Vol. 1. Prentice Hall Upper Saddle River, 2002.

[24] K. Schwaber. "Scrum development process". In: *Business object design and implementation*. Springer, 1997, pp. 117–134.

[25] K. Schwaber. *Agile project management with Scrum*. Microsoft press, 2004.

[26] J. K. Liker. *Toyota way: 14 management principles from the world's greatest manufacturer*. McGraw-Hill Education, 2004.

[27] M. Poppendieck and T. Poppendieck. *Lean software development: an agile toolkit*. Addison-Wesley, 2003.

[28] T. Dingsøyr and N. B. Moe. "Research challenges in large-scale agile software development". In: *ACM SIGSOFT Software Engineering Notes* 38.5 (2013), pp. 38–39.

[29] M. Paasivaara, S. Durasiewicz, and C. Lassenius. "Using scrum in a globally distributed project: a case study". In: *Software Process: Improvement and Practice* 13.6 (2008), pp. 527–544.

[30] E. Moore and J. Spens. "Scaling agile: Finding your agile tribe". In: *Agile 2008 Conference*. IEEE. 2008, pp. 121–124.

[31] T. Dingsøyr, T. E. Fægri, and J. Itkonen. "What is large in large-scale? A taxonomy of scale for agile software development". In: *International Conference on Product-Focused Software Process Improvement*. Springer. 2014, pp. 273–276.

[32] K. Petersen and C. Wohlin. "The effect of moving from a plan-driven to an incremental software development approach with agile practices". In: *Empirical Software Engineering* 15.6 (2010), pp. 654–693.

[33] E. Bjarnason, K. Wnuk, and B. Regnell. "A case study on benefits and side-effects of agile practices in large-scale requirements engineering". In: *proceedings of the 1st workshop on agile requirements engineering*. 2011, pp. 1–5.

[34] H. Berger and P. Beynon-Davies. "The utility of rapid application development in large-scale, complex projects". In: *Information Systems Journal* 19.6 (2009), pp. 549–570.

[35] T. Dingsøyr and N. B. Moe. "Towards principles of large-scale agile development". In: *International Conference on Agile Software Development*. Springer. 2014, pp. 1–8.

[36] C. R. Lincoln. "Best practices in software measurement: How to use metrics to improve project and process performance". In: *Software Quality Professional* 7.3 (2005), p. 48.

[37] N. Fenton. "Software measurement: A necessary scientific basis". In: *IEEE Transactions on software engineering* 20.3 (1994), pp. 199–206.

[38] S. Brocklehurst, P. Chan, B. Littlewood, and J. Snell. "Recalibrating software reliability models". In: *IEEE Transactions on Software Engineering* 16.4 (1990), pp. 458–470.

[39] V. R. Basili, M. Lindvall, M. Regardie, C. Seaman, J. Heidrich, J. Münch, D. Rombach, and A. Trendowicz. "Linking software development and business strategy through measurement". In: *Computer* 43.4 (2010), pp. 57–65.

[40] V. R. Basili and D. M. Weiss. "A methodology for collecting valid software engineering data". In: *IEEE Transactions on software engineering* 6 (1984), pp. 728–738.

[41] G. C. Victor R Basili and H. D. Rombach. "The goal question metric approach". In: *Encyclopedia of software engineering* (1994), pp. 528–532.

[42] B. Kitchenham. "What's up with software metrics?–A preliminary mapping study". In: *Journal of systems and software* 83.1 (2010), pp. 37–51.

[43] O. Gómez, H. Oktaba, M. Piattini, and F. Garcıa. "A systematic review measurement in software engineering: State-of-the-art in measures". In: *International Conference on Software and Data Technologies*. Springer. 2006, pp. 165–176.

[44] B. Kitchenham. "Procedures for performing systematic reviews". In: *Keele, UK, Keele University* 33.2004 (2004), pp. 1–26.

[45] C. Patel, M. Lycett, R. Macredie, and S. de Cesare. "Perceptions of agility and collaboration in software development practice". In: *Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06)*. Vol. 1. IEEE. 2006, pp. 10c–10c.

[46] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson. "Systematic mapping studies in software engineering". In: *12th International Conference on Evaluation and Assessment in Software Engineering (EASE) 12*. 2008, pp. 1–10.

[47] Software, S. E. S. Committee, et al. "IEEE Standard Adoption of ISO/IEC 15939: 2007 Systems and Software Engineering Measurement Process". In: *Electronic book]. IEEE Computer Society* (2008).

[48] M. Poppendieck and T. Poppendieck. *Implementing lean software development: from concept to cash*. Pearson Education, 2007.

[49] S. L. Ramırez-Mora and H. Oktaba. "Productivity in agile software development: a systematic mapping study". In: *2017 5th international conference in software engineering research and innovation (CONISOFT)*. IEEE. 2017, pp. 44–53.

[50] B. A. Kitchenham, D. Budgen, and P. Brereton. *Evidence-based software engineering and systematic reviews*. Vol. 4. CRC press, 2015.

[51] S. M. A. Shah, E. Papatheocharous, and J. Nyfjord. "Measuring productivity in agile software development process: a scoping study". In: *Proceedings of the 2015 International Conference on Software and System Process*. 2015, pp. 102–106.

[52] H. Arksey and L. O'Malley. "Scoping studies: towards a methodological framework". In: *International journal of social research methodology* 8.1 (2005), pp. 19–32.

[53] F. Kišš and B. Rossi. "Agile to lean software development transformation: A systematic literature review". In: *2018 Federated Conference on Computer Science and Information Systems (FedCSIS)*. IEEE. 2018, pp. 969–973.

[54] K. Wnuk and K. C. Maddila. "Agile and lean metrics associated with requirements engineering". In: *Proceedings of the 27th International Workshop on Software Measurement and 12th International Conference on Software Process and Product Measurement*. 2017, pp. 33–40.

[55] W. H. C. Almeida, F. Furtado, L. de Aguiar Monteiro, F. Escobar, and S. K. G. e Silva. "Systematic Review on the Use of Metrics for Estimating the Effort and Cost of Software Applicable to the Brazilian Public Sector". In: *ICSEA 2020* (2020), p. 43.

[56] R. Kurnia, R. Ferdiana, and S. Wibirama. "Software metrics classification for agile scrum process: a literature review". In: *2018 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*. IEEE. 2018, pp. 174–179.

[57] D. Mishra and S. Abdalhamid. "Software Quality Issues in SCRUM: A Systematic Mapping." In: *Journal of Universal Computer Science* 24.12 (2018), pp. 1690–1716.

[58] Ö. Uludağ, M. Kleehaus, X. Xu, and F. Matthes. "Investigating the role of architects in scaling agile frameworks". In: *2017 IEEE 21st International Enterprise Distributed Object Computing Conference (EDOC)*. IEEE. 2017, pp. 123–132.

[59] P. Rodrıguez, A. Haghighatkhah, L. E. Lwakatare, S. Teppola, T. Suomalainen, J. Eskeli, T. Karvonen, P. Kuvaja, J. M. Verner, and M. Oivo. "Continuous deployment of software intensive products and services: A systematic mapping study". In: *Journal of Systems and Software* 123 (2017), pp. 263–291.

[60] D. Leffingwell. *SAFe 5.0 Framework*. Feb. 2021. URL: `https://www.scaledagileframework.com/metrics/`.

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

[62] A. Poligadu and R. K. Moloo. "An innovative measurement programme for agile governance". In: *International Journal of Agile Systems and Management* 7.1 (2014), pp. 26–60.

Primary Sources

[S1] D. R. Greening. "Enterprise Scrum: Scaling Scrum to the Executive Level". In: *2010 43rd Hawaii International Conference on System Sciences*, pp. 1–10. ISBN: 1530-1605. DOI: 10.1109/HICSS.2010.186.

[S2] D. R. Greening. "Release duration and enterprise agility". In: *2013 46th Hawaii International Conference on System Sciences*. IEEE, pp. 4835–4841. ISBN: 1467359335.

[S3] P. S. M. dos Santos, A. Varella, C. R. Dantas, and D. B. Borges. "Visualizing and managing technical debt in agile development: An experience report". In: *International Conference on Agile Software Development*. Springer, pp. 121–134.

[S4] P. Grimaldi, L. Perrotta, V. Corvello, and S. Verteramo. "An agile, measurable and scalable approach to deliver software applications in a large enterprise". In: *International Journal of Agile Systems and Management* 9.4 (2016), pp. 326–339. ISSN: 1741-9174.

[S5] M. Kajko-Mattsson and J. Nyfjord. "A model of agile evolution and maintenance process". In: *2009 42nd Hawaii International Conference on System Sciences*. IEEE, pp. 1–10. ISBN: 0769534503.

[S6] B. Horlach, I. Schirmer, and P. Drews. "Agile portfolio Management: Design Goals and Principles". In: *ECIS*.

[S7] H. Smits and K. Rilliet. "Agile experience report: Transition and complexity at cisco voice technology group". In: *2011 Agile Conference*. IEEE. 2011, pp. 274–278.

[S8] I. Flaounas and A. Friedman. "Bridging the gap between business, design and product metrics". In: *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*. 2019, pp. 1–6.

[S9] V. S. Ida Korpivaara Tuure Tuunanen. "Performance Measurement in Scaled Agile Organizations". In: *Hawaii International Conference on System Sciences*. DOI: 10.24251/HICSS.2021.830.

[S10] M. Staron and W. Meding. "Factors Determining Long-term Success of a Measurement Program: An Industrial Case Study". In: *e Informatica Softw. Eng. J.* 5 (2011), pp. 7–23.

[S11] A. Tengshe and S. Noble. "Establishing the agile PMO: Managing variability across projects and portfolios". In: *Agile 2007 (AGILE 2007)*. IEEE, pp. 188–193. ISBN: 0769528724.

[S12] C. J. Stettina and L. Schoemaker. "Reporting in agile portfolio management: routines, metrics and artefacts to maintain an effective oversight". In: *International Conference on Agile Software Development*. Springer, Cham, pp. 199–215.

[S13] I. Smeekes, H. Borgman, and H. Heier. "A Wheelbarrow Full of Frogs: Understanding Portfolio Management for Agile Projects". In: *Proceedings of the 51st Hawaii International Conference on System Sciences*. ISBN: 0998133116.

[S14] H. Samios. "Overcoming Traditional Project Release Reporting with an Agile Approach Focused on Change". In: *2012 Agile Conference*, pp. 131–135. DOI: 10.1109/Agile.2012.31.

[S15] B. Snyder and B. Curtis. "Using Analytics to Guide Improvement during an Agile–DevOps Transformation". In: *IEEE Software* 35.1 (2018), pp. 78–83. ISSN: 1937-4194. DOI: 10.1109/MS.2017.4541032.

[S16] R. Tabib. "Need 4 speed: leverage new metrics to boost your velocity without compromising on quality". In: *2013 Agile Conference*. IEEE, pp. 117–120. ISBN: 0769550762.

[S17] J. Sutherland, A. Viktorov, J. Blount, and N. Puntikov. "Distributed Scrum: Agile Project Management with Outsourced Development Teams". In: *2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07)*, 274a–274a. ISBN: 1530-1605. DOI: 10.1109/HICSS.2007.180.

[S18] K. Petersen and C. Wohlin. "The effect of moving from a plan-driven to an incremental software development approach with agile practices". In: *Empirical Software Engineering* 15.6 (2010), pp. 654–693. ISSN: 1573-7616.

[S19] P. Ram, P. Rodriguez, and M. Oivo. "Software process measurement and related challenges in agile software development: A multiple case study". In: *International Conference on Product-Focused Software Process Improvement*. Springer, pp. 272–287.

[S20] M. Staron and W. Meding. "Monitoring bottlenecks in agile and lean software development projects–a method and its industrial use". In: *International Conference on Product Focused Software Process Improvement*. Springer, pp. 3–16.

[S21] D. I. Sjøberg, A. Johnsen, and J. Solberg. "Quantifying the effect of using kanban versus scrum: A case study". In: *IEEE software* 29.5 (2012), pp. 47–53. ISSN: 0740-7459.

[S22] S. Matthiesen and P. Bjørn. "When distribution of tasks and skills are fundamentally problematic: A failure story from global software outsourcing". In: *Proceedings of the ACM on Human-Computer Interaction* 1.CSCW (2017), pp. 1–16. ISSN: 2573-0142.

[S23] R. K. Gupta, P. Manikreddy, S. Naik, and K. Arya. "Pragmatic approach for managing technical debt in legacy software project". In: *Proceedings of the 9th India Software Engineering Conference*, pp. 170–176.

[S24] T. Gustavsson. "Visualizing Inter-Team Coordination". In: *Proceedings of the Evaluation and Assessment in Software Engineering*. 2020, pp. 306–311.

[S25] S. Gruschwitz and F. Schlosser. "Towards an integrated model for managing product and process quality in agile software projects". In: *7th International Research Workshop on Information Technology Project Management (IRWITPM 2012)*, pp. 147–155.

[S26] Y. Dubinsky, D. Talby, O. Hazzan, and A. Keren. "Agile metrics at the israeli air force". In: *Agile Development Conference (ADC'05)*. IEEE, pp. 12–19. ISBN: 0769524877.

[S27] O. Erdoğan, M. E. Pekkaya, and H. Gök. "More effective sprint retrospective with statistical analysis". In: *Journal of Software: Evolution and Process* 30.5 (2018), e1933. ISSN: 2047-7473.

[S28] D. Talby, O. Hazzan, Y. Dubinsky, and A. Keren. "Reflections on reflection in agile software development". In: *AGILE 2006 (AGILE'06)*, 11 pp.–112. DOI: 10.1109/AGILE.2006.45.

[S29] P. Bossi. "Using actual time: learning how to estimate". In: *International Conference on Extreme Programming and Agile Processes in Software Engineering*. Springer, pp. 244–253.

[S30] S. Goeschl, M. Herp, and C. Wais. "When agile meets OO testing: a case study". In: *Proceedings of the 1st Workshop on Testing Object-Oriented Systems*, pp. 1–5.

[S31] A. Janus, R. Dumke, A. Schmietendorf, and J. Jäger. "The 3c approach for agile quality assurance". In: *2012 3rd International Workshop on Emerging Trends in Software Metrics (WETSoM)*. IEEE, pp. 9–13. ISBN: 1467317624.

[S32] J. Diebold, P. Diebold, and A. Vetter. "Agile Meets Assessments: Case Study on How to Do Agile Process Improvement in a Very Small Enterprise". In: *International Conference on Product-Focused Software Process Improvement*. Springer, pp. 31–47.

[S33] C. J. Poole, T. Murphy, J. W. Huisman, and A. Higgins. "Extreme maintenance". In: *Proceedings IEEE International Conference on Software Maintenance. ICSM 2001*. IEEE, pp. 301–309. ISBN: 0769511899.

[S34] J. Downs, J. Hosking, and B. Plimmer. "Status communication in agile software teams: A case study". In: *2010 Fifth International Conference on Software Engineering Advances*. IEEE, pp. 82–87. ISBN: 1424477883.

[S35] W. Meding. "Effective monitoring of progress of agile software development teams in modern software companies: an industrial case study". In: *Proceedings of the 27th International Workshop on Software Measurement and 12th International Conference on Software Process and Product Measurement*. 2017, pp. 23–32.

[S36] A. Friedman and I. Flaounas. "The right metric for the right stakeholder: a case study of improving product usability". In: *Proceedings of the 30th Australian Conference on Computer-Human Interaction*, pp. 602–606.

[S37] C. T. Wolf and J. L. Blomberg. "Ambitions and Ambivalences in Participatory Design: Lessons from a Smart Workplace Project". In: *Proceedings of the 16th Participatory Design Conference 2020-Participation (s) Otherwise-Volume 1*, pp. 193–202.

[S38] M. Staron, W. Meding, and P. Baniasad. "Information Needs for SAFe Teams and Release Train Management: A Design Science Research Study". In: *IWSM-Mensura*.

[S39] H. Alahyari, R. B. Svensson, and T. Gorschek. "A study of value in agile software development organizations". In: *Journal of Systems and Software* 125 (2017), pp. 271–288. ISSN: 0164-1212.

[S40] A. N. Abdel-Hamid. "Refactoring as a lifeline: Lessons learned from refactoring". In: *2013 Agile Conference*. IEEE, pp. 129–136. ISBN: 0769550762.

[S41] P. Ram, P. Rodriguez, M. Oivo, and S. Martínez-Fernández. "Success factors for effective process metrics operationalization in agile software development: A multiple case study". In: *2019 IEEE/ACM International Conference on Software and System Processes (ICSSP)*. IEEE, pp. 14–23. ISBN: 1728133939.

[S42] F. Fagerholm, M. Ikonen, P. Kettunen, J. Münch, V. Roto, and P. Abrahamsson. "Performance Alignment Work: How software developers experience the continuous adaptation of team performance in Lean and Agile environments". In: *Information and Software Technology* 64 (2015), pp. 132–147. ISSN: 0950-5849.

[S43] P. Kettunen, M. Laanti, F. Fagerholm, and T. Mikkonen. "Agile in the Era of Digitalization: A Finnish Survey Study". In: *Product-Focused Software Process Improvement*. Ed. by X. Franch, T. Männistö, and S. Martínez-Fernández. Springer International Publishing, pp. 383–398. ISBN: 978-3-030-35333-9.

[S44] J. Garzás and M. C. Paulk. "A case study of software process improvement with CMMI-DEV and Scrum in Spanish companies". In: *Journal of Software: Evolution and Process* 25.12 (2013), pp. 1325–1333. ISSN: 2047-7473.

[S45] M. K. Lárusdóttir, Å. Cajander, and M. Simader. "Continuous Improvement in Agile Development Practice". In: *Human-Centered Software Engineering*. Ed. by S. Sauer, C. Bogdan, P. Forbrig, R. Bernhaupt, and M. Winckler. Springer Berlin Heidelberg, pp. 57–72. ISBN: 978-3-662-44811-3.

[S46] F. Sambinelli and M. A. F. Borges. "Survey on Strategies to Increase Customer Value in Brazilian Agile Software Development Companies". In: *2019 14th Iberian Conference on Information Systems and Technologies (CISTI)*. IEEE, pp. 1–7. ISBN: 9899843490.

[S47] B. Eckfeldt, R. Madden, and J. Horowitz. "Selling agile: target-cost contracts". In: *Agile Development Conference (ADC'05)*. IEEE, pp. 160–166. ISBN: 0769524877.

[S48] P. Green. "Measuring the impact of scrum on product development at adobe systems". In: *2011 44th Hawaii International Conference on System Sciences*. IEEE, pp. 1–10. ISBN: 1424496187.

[S49] F. Kortum, J. Klünder, O. Karras, W. Brunotte, and K. Schneider. "Which Information Help agile Teams the Most? An Experience Report on the Problems and Needs". In: *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, pp. 306–313. ISBN: 1728195322.

[S50] P. E. McMahon. "Are the Right People Measuring the Right Things? A Lean Path to Achieving Business Objectives". In: *CrossTalk* 21 (2008), pp. 16–20.

[S51] V. Augustine, J. Hudepohl, P. Marcinczak, and W. Snipes. "Deploying software team analytics in a multinational organization". In: *IEEE Software* 35.1 (2017), pp. 72–76. ISSN: 0740-7459.

[S52] N. Oza and M. Korkala. "Lessons Learned In Implementing Agile Software Development Metrics". In: *UKAIS*.

[S53] L. Siddique and B. A. Hussein. “A qualitative study of success criteria in Norwegian agile software projects from suppliers’ perspective”. In: (2016). ISSN: 2182-7796.

[S54] P. Gregory, L. Barroca, H. Sharp, A. Deshpande, and K. Taylor. “The challenges that challenge: Engaging with agile practitioners’ concerns”. In: *Information and Software Technology* 77 (2016), pp. 92–104. ISSN: 0950-5849.

[S55] J. Chen, J. Xiao, Q. Wang, L. J. Osterweil, and M. Li. “Refactoring planning and practice in agile software development: an empirical study”. In: *Proceedings of the 2014 International Conference on Software and System Process*, pp. 55–64.

[S56] M. Pacheco, A.-L. Mesquida, and A. Mas. “Being Agile While Coaching Teams Using Their Own Data”. In: *Systems, Software and Services Process Improvement*. Ed. by X. Larrucea, I. Santamaria, R. V. O’Connor, and R. Messnarz. Springer International Publishing, pp. 426–436. ISBN: 978-3-319-97925-0.

[S57] S. Hassani-Alaoui, A.-F. Cameron, and T. Giannelia. ““We use scrum, but...”: Agile modifications and project success”. In: *Proceedings of the 53rd Hawaii International Conference on System Sciences*. ISBN: 0998133132.

[S58] D. J. Anderson. “Stretching agile to fit CMMI level 3-the story of creating MSF for CMMI/spl reg/process improvement at Microsoft corporation”. In: *Agile Development Conference (ADC’05)*. IEEE, pp. 193–201. ISBN: 0769524877.

[S59] R. Hekkala, M.-K. Stein, M. Rossi, and K. Smolander. “Challenges in Transitioning to an Agile Way of Working”. In: *Hawaii International Conference on System Sciences*. DOI: 10.24251/HICSS.2017.707.

[S60] A. Elssamadisy and G. Schalliol. “Recognizing and responding to" bad smells" in extreme programming”. In: *Proceedings of the 24th International conference on Software Engineering*, pp. 617–622.

[S61] A. Fabijan, H. H. Olsson, and J. Bosch. “The lack of sharing of customer data in large software organizations: challenges and implications”. In: *International Conference on Agile Software Development*. Springer, pp. 39–52.

[S62] I. Lehto and K. Rautiainen. “Software development governance challenges of a middle-sized company in agile transition”. In: *2009 ICSE Workshop on Software Development Governance*. IEEE, pp. 36–39. ISBN: 1424437369.

[S63] P. Hodgkins and L. Hohmann. “Agile program management: Lessons learned from the verisign managed security services team”. In: *Agile 2007 (AGILE 2007)*. IEEE, pp. 194–199. ISBN: 0769528724.

[S64] R. Polk. “Agile and Kanban in coordination”. In: *2011 Agile Conference*. IEEE, pp. 263–268. ISBN: 161284426X.

[S65] A. Salameh and J. M. Bass. “Heterogeneous tailoring approach using the spotify model”. In: *Proceedings of the Evaluation and Assessment in Software Engineering*. 2020, pp. 293–298.

[S66] S. Pradhan and V. Nanniyur. "Large scale quality transformation in hybrid development organizations–A case study". In: *Journal of Systems and Software* 171 (2021), p. 110836.

[S67] H. Koehnemann and M. Coats. "Experiences Applying Agile Practices to Large Systems". In: *2009 Agile Conference*, pp. 295–300. DOI: 10.1109/AGILE.2009.59.

[S68] T. Cheng, S. Jansen, and M. Remmers. "Controlling and monitoring agile software development in three dutch product software companies". In: *2009 ICSE Workshop on Software Development Governance*, pp. 29–35. DOI: 10.1109/SDG.2009.5071334.

[S69] R. Sirkiä and M. Laanti. "Adaptive Finance and Control: Combining Lean, Agile, and Beyond Budgeting for Financial and Organizational Flexibility". In: *2015 48th Hawaii International Conference on System Sciences*, pp. 5030–5037. ISBN: 1530-1605. DOI: 10.1109/HICSS.2015.596.

[S70] S. Coyle and J. Barata. "Socially-Constructed Metrics for Agile Quality: An Action-Research Study". In:

[S71] K. Korhonen. "Migrating defect management from waterfall to agile software development in a large-scale multi-site organization: A case study". In: *International Conference on Agile Processes and Extreme Programming in Software Engineering*. Springer, pp. 73–82.

[S72] D. Talby and Y. Dubinsky. "Governance of an Agile Software Project". In: *Proceedings of the 2009 ICSE Workshop on Software Development Governance, SDG 2009* (2009). DOI: 10.1109/SDG.2009.5071336.

[S73] K. J. Padmini, H. D. Bandara, and I. Perera. "Use of software metrics in agile software development process". In: *2015 Moratuwa Engineering Research Conference (MERCon)*. IEEE, pp. 312–317. ISBN: 1479917400.

[S74] T. Winz, S. Streubel, C. Tancau, and S. Dhone. "Clean A-SPICE Processes and Agile Methods Are the Key to Modern Automotive Software Engineering". In: *Systems, Software and Services Process Improvement*. Ed. by M. Yilmaz, J. Niemann, P. Clarke, and R. Messnarz. Springer International Publishing, pp. 571–586. ISBN: 978-3-030-56441-4.

[S75] N. Tekin, M. Kosa, M. Yilmaz, P. Clarke, and V. Garousi. "Visualization, Monitoring and Control Techniques for Use in Scrum Software Development: An Analytic Hierarchy Process Approach". In: *European Conference on Software Process Improvement*. Springer, pp. 45–57.

[S76] L. Layman, L. Williams, D. Damian, and H. Bures. "Essential communication practices for Extreme Programming in a global software development team". In: *Inf. Softw. Technol.* 48 (2006), pp. 781–794.

[S77] C. R. Jakobsen and T. Poppendieck. "Lean as a scrum troubleshooter". In: *2011 Agile Conference*. IEEE, pp. 168–174. ISBN: 161284426X.

[S78] L. Cao and E. H. Park. "Understanding Goal-Directed Emotions in Agile Software Development Teams". In: (2017).

[S79] M. C. Marinovici, H. Kirkham, K. A. Glass, and L. C. Carlsen. "Engineering quality while embracing change: Lessons learned". In: *2013 46th Hawaii International Conference on System Sciences*. IEEE, pp. 4810–4816. ISBN: 1467359335.

[S80] T. Hacaloglu and O. Demirors. "Measureability of functional size in Agile software projects: Multiple case studies with COSMIC FSM". In: *2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, pp. 204–211. ISBN: 1728134218.

[S81] D. Batra, W. Xia, D. VanderMeer, and K. Dutta. "Balancing agile and structured development approaches to successfully manage large distributed software projects: A case study from the cruise line industry". In: *Communications of the Association for Information Systems* 27.1 (2010), p. 21. ISSN: 1529-3181.

[S82] S. Modi, P. Abbott, and S. Counsell. "Negotiating common ground in distributed agile development: A case study perspective". In: *2013 IEEE 8th International Conference on Global Software Engineering*. IEEE, pp. 80–89. ISBN: 0769550576.

[S83] K. Korhonen. "Evaluating the impact of an agile transformation: a longitudinal case study in a distributed context". In: *Software Quality Journal* 21.4 (2013), pp. 599–624. ISSN: 1573-1367.

[S84] C. Fuchs. "Adapting (to) Agile Methods: Exploring the Interplay of Agile Methods and Organizational Features". In: *Proceedings of the 52nd Hawaii International Conference on System Sciences*. 2019.

[S85] P. Hodgetts. "Refactoring the development process: Experiences with the incremental adoption of agile practices". In: *Agile Development Conference*. IEEE, pp. 106–113. ISBN: 0769522483.

[S86] H. Klein and S. Canditt. "Using opinion polls to help measure business impact in agile development". In: *proceedings of the 1st international workshop on business impact of process improvements*, pp. 25–32.

[S87] N. Abbas, A. M. Gravell, and G. B. Wills. "The impact of organization, project and governance variables on software quality and project success". In: *2010 Agile Conference*. IEEE, pp. 77–86. ISBN: 142447731X.

[S88] W. Trumler and F. Paulisch. "How "Specification by Example" and test-driven development help to avoid technial debt". In: *2016 IEEE 8th International Workshop on Managing Technical Debt (MTD)*. IEEE, pp. 1–8. ISBN: 150903854X.

[S89] V. Trapa and S. Rao. "T3-tool for monitoring agile development". In: *AGILE 2006 (AGILE'06)*. IEEE, 6 pp.–248. ISBN: 0769525628.

[S90] Y. Wang, M. V. Mäntylä, S. Demeyer, K. Wiklund, S. Eldh, and T. Kairi. "Software Test Automation Maturity–A Survey of the State of the Practice". In: *arXiv preprint arXiv:2004.09210* (2020).

[S91] X. Zhao, X. Xuan, A. Wang, D. Liu, and L. Zheng. "Software Quality Control via Exit Criteria Methodology: An Industrial Experience Report". In: *2014 21st Asia-Pacific Software Engineering Conference*. Vol. 2. IEEE, pp. 23–26. ISBN: 1479974269.

[S92] A. Elbanna and D. Murray. *Organizing Projects for Innovation: A Collective Mindfulness Perspective*. 2009, p. 276.

[S93] P. Rachow, S. Schröder, and M. Riebisch. "Missing Clean Code Acceptance and Support in Practice - An Empirical Study". In: *2018 25th Australasian Software Engineering Conference (ASWEC)*, pp. 131–140. ISBN: 2377-5408. DOI: 10.1109/ASWEC.2018.00026.

[S94] R. Berntsson Svensson. "Measuring Team Innovativeness: A Multiple Case Study of Agile and Lean Software Developing Companies". In: Product-Focused Software Process Improvement. Springer International Publishing, pp. 37–51. ISBN: 978-3-319-69926-4.

[S95] H. Edison, X. Wang, and P. Abrahamsson. "Lean startup: why large software companies should care". In: *Scientific Workshop Proceedings of the XP2015* (2015).

[S96] M. Seikola, H. Loisa, and A. Jagos. "Kanban Implementation in a Telecom Product Maintenance". In: *2011 37th EUROMICRO Conference on Software Engineering and Advanced Applications*, pp. 321–329. ISBN: 2376-9505. DOI: 10.1109/SEAA.2011.56.

[S97] N. Zabkar, T. Hovelja, J. Urevc, and V. Mahnic. "Introducing agile software development: lessons learned from the first scrum project in a Slovenian company". In:

[S98] M. Kuniavsky and S. Raghavan. "Guidelines are a tool: building a design knowledge management system for programmers". In: (2005).

[S99] S. L. Ramírez-Mora, H. Oktaba, and J. Patlán Pérez. "Group maturity, team efficiency, and team effectiveness in software development: A case study in a CMMI-DEV Level 5 organization". In: *Journal of Software: Evolution and Process* 32.4 (2020), e2232. ISSN: 2047-7473. DOI: https://doi.org/10.1002/smr.2232. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/smr.2232.

[S100] N. Schooenderwoert. *Embedded Agile Project by the Numbers With Newbies*. 2006, pp. 351–366. DOI: 10.1109/AGILE.2006.24.

[S101] A. Rendell. "Effective and Pragmatic Test Driven Development". In: *Agile 2008 Conference*, pp. 298–303. DOI: 10.1109/Agile.2008.45.

[S102] N. Davis. "Driving Quality Improvement and Reducing Technical Debt with the Definition of Done". In: *2013 Agile Conference*, pp. 164–168. DOI: 10.1109/AGILE.2013.21.

[S103] P. J. Guinan, S. Parise, and N. Langowitz. "Creating an innovative digital project team: Levers to enable digital transformation". In: *Business Horizons* 62.6 (2019), pp. 717–727. URL: https://EconPapers.repec.org/RePEc:eee:bushor:v:62:y:2019:i:6:p:717-727.

[S104] K. Long and D. Starr. "Agile Supports Improved Culture and Quality for Healthwise". In: *Agile 2008 Conference*, pp. 160–165. DOI: 10.1109/Agile.2008.61.

[S105] K. Conboy, M. Pikkarainen, and X. Wang. "Agile Practices in Use from an Innovation Assimilation Perspective: A Multiple Case Study". In: *ICIS*.

[S106] G. M. Roche and Jr. "Agile Portfolio Management at NYSE". In: *2012 Agile Conference*, pp. 117–122. DOI: 10.1109/Agile.2012.12.

[S107] M. Lárusdóttir, Å. Cajander, and J. Gulliksen. "Informal feedback rather than performance measurements – user-centred evaluation in Scrum projects". In: *Behav. Inf. Technol.* 33.11 (2014), pp. 1118–1135. ISSN: 0144-929X. DOI: 10.1080/0144929x.2013.857430. URL: https://doi.org/10.1080/0144929X.2013.857430.

[S108] A. Anwar, A. A. Kamel, and E. Ahmed. "Agile adoption case study, pains, challenges & benefits". In: *Proceedings of the 2nd Africa and Middle East Conference on Software Engineering*. 2016, pp. 60–65.

[S109] B. Schatz and I. Abdelshafi. "Primavera Gets Agile: A Successful Transition to Agile Development". In: *IEEE Softw.* 22.3 (2005), pp. 36–42. ISSN: 0740-7459. DOI: 10.1109/ms.2005.74. URL: https://doi.org/10.1109/MS.2005.74.

[S110] D. R. Greening. "Agile Enterprise Metrics". In: *2015 48th Hawaii International Conference on System Sciences*, pp. 5038–5044. ISBN: 1530-1605. DOI: 10.1109/HICSS.2015.597.