DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Information Systems

Investigating the Adoption and Management of Metrics in Large-Scale Agile Software Development at a German IT-Provider

Leon Marius Menzel

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Information Systems

Investigating the Adoption and Management of Metrics in Large-Scale Agile Software Development at a German IT-Provider

Untersuchung der Einführung und des Managements von Metriken in der groß angelegten agilen Softwareentwicklung bei einem deutschen IT-Anbieter

Author:	Leon Marius Menzel
Supervisor:	Prof. Dr. Florian Matthes
Advisor:	Pascal Maurice Philipp, M.Sc.
Submission Date:	15.09.2021

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

Munich, 15.09.2021

Leon Marius Menzel

Abstract

Over the last two decades, agile methods have become the predominant software development mode [106]. Originally developed for use in small, co-located single teams, companies began to adopt agile methods on a larger scale [45]. Frameworks, such as the Scaled Agile Framework or Scrum-at-Scale, help to organize team collaboration within the large-scale agile software development environment. In order to continue to facilitate transparency and control even over larger team and process structures, the need arose to introduce metrics and integrate them into the workflows of the agile frameworks. Several researchers have already studied the introduction of metrics in agile single-team settings [13, 18, 22]. However, an investigation of the introduction of metrics in the large-scale agile environment considering the agile scaling frameworks is missing so far [27]. Likewise, a research gap concerns the long-term management of metrics and the associated metrics teams within a holistic reference framework [73, 83]. Hence, the objective of this master's thesis was to investigate the current state of the metrics usage in relation to their associated goals within a case organization to build and evaluate a systematic approach for metrics adoption and long-term management. As part of an action design research, the approach was executed in two analysis units of the case organization and subsequently evaluated through two surveys. The case study results indicate a correlation between the nature of current metrics use and the level of adoption of agile practices and scaled frameworks. The main contribution of this thesis lies in the design of an approach for the goal-based adoption and management of metrics in scaled agile software development. The thesis also describes the execution of the approach that led the case organization to adopt thirteen and nineteen goal-based metrics, respectively, and to uncover their improvement potential in metrics management. In addition, the case organization received specific recommendations for the introduction and management of metrics through the approach. Survey results show a high level of approval and, thus, confirm the practical relevance of the different steps and components of the approach.

Contents

List of Figures

List of Tables

Acronyms

AI Artificial Intelligence. 9

ART Agile Release Train. ix, 18–20, 31, 42, 44–48, 50–52, 54, 56, 64, 73, 86–90, 92–98, 100, 101, 118, 120

CD Continuous Deployment. 19

CE Continuous Exploration. 19

CI Continuous Integration. 19

CPO Chief Product Owner. 22, 23

CTO Chief Technology Officer. 44

DAD Discipline Agile Delivery. 16, 17, 29, 30

DevOps software development and IT operations. 9, 17, 19

DSDM Dynamic System Development Method. 8, 16

EAT Executive Action Team. 23

EMS Executive MetaScrum. 23

GQM Goal Question Metric. vii, 3–5, 32–34, 37, 59–61, 64, 65, 67, 68, 75–79, 88–91, 93, 103–105, 108, 109, 119, 124

IoT Internet of Things. 9

IS Information Systems. 5

KPI Key Performance Indicator. 24, 25, 38

LeSS Large-Scale Scrum. 16, 17

MMF Minimal Marketable Feature. 32

MVB Minimum Viable Bureaucracy. 21, 23

MVP Minimal Viable Product. 32

NPS Net Present Value. 30

OKR Objectives and Key Results. 40

PI Product Increment. 18–20, 30–32, 50–52, 54, 86, 90–96, 99, 100

PO Product Owner. 13, 22, 23, 44, 50, 53, 54, 56, 118, 120

RAGE Recipes for Agile Governance in the Enterprise. 16, 17

RI Result Indicator. 25

RTE Release Train Engineer. 20, 35, 44, 50, 54–56, 86, 93, 95, 96, 99, 120

SAFe Scaled Agile Framework. vii, 15–21, 24, 29–32, 35, 39, 40, 46, 51, 54, 62, 68, 71, 90, 91, 93, 118, 120

SLA Service-Level-Agreements. 54

SM Scrum Master. 13, 23, 44, 118, 120

SoS Scrum of Scrums. 21–23

SoSM Scrum of Scrums Master. 21–23

SoSoS Scrum of Scrum of Scrums. 23

SPC SAFe Program Consultants. 18

XP Extreme Programming. 8, 16–18

1. Introduction

This chapter explains in Section 1.1 why adopting and managing metrics, and strategic goals in large-scale agile software development is a relevant, new, and interesting research endeavor. Additionally, it presents in Section 1.2 the primary research objectives and the related guiding research questions of this thesis. Section 1.3 provides an overview of the research methodologies used to answer the research questions.

1.1. Motivation

The following section introduces the rise of large-scale agile software development practices to understand why large-scale agile software development metrics are an exciting and significant topic. Subsequently, the section continues with the current use of metrics in software development, their application areas, challenges, and success factors. Finally, the aim of the thesis and the related intention to close the existing research gap follows.

The rise of large-scale agile software development practices

Agile software development methodologies and practices have spread across the industry since the publication of the Agile Manifesto in 2001 and have become a quasi-standard. This development is borne out by the figures from the "14th State of Agile Survey", according to which 95% of respondents state that their organization uses agile development methodologies. The most popular methodology is certainly Scrum, which either alone or in combination with other methodologies has a market share of over 75% [106]. The success of agile methodologies and practices has led to an extension of its original scope from small, on-site, single-team deployments to larger, distributed, cross-team, and company-wide contexts [2, 93]. Compared to smaller setups, ideal for agile software development, this evolution poses challenges for organizations when adopting and scaling agile. On the one hand, the scaling of agile teams results in cross-team dependencies, duplication of work, and communication overhead. On the other hand, classic management structures suffer from limitations in reacting effectively to changing business requirements and priorities in scaled organizational structures [25]. Overall, both issues can lead to a reduction in the volume, speed, and quality of software products per team, thus compromising the agility of an organization [110]. Therefore, about a decade ago, large-scale frameworks began to be developed and established in organizations with the goal of structuring and aligning collaboration and communication between teams. These frameworks also aim to make larger team structures and processes transparent and easier to control. In addition, the frameworks enable the holistic integration of the software development life-cycle with the project and portfolio management [4]. Nevertheless, an overwhelming majority of 84% of organizations signal that their agile practices are at a low level of maturity, leaving room for further improvement and exploration of the adoption of agile initiatives [106].

Metrics in software development

As part of large-scale agile frameworks, metrics serve to fuel data-driven decision-making and streamline the work of teams, stakeholders, and leadership [110]. In general, metrics have been used as an essential component of software development for decades. For example, metrics measure consistency and completeness of requirements, testing effort and errors, and team size and productivity [35]. Probably one of the most famous motivations for using metrics as a measurement instrument comes from the British mathematician and engineer Lord Kelvin who said: "To measure is to know", or "If you cannot measure it, you cannot improve it." Behind these phrases is the idea of gaining transparency and control about action to change it for the better. The most successful companies (such as Microsoft, IBM, and AT&T) excel in control and transparency over business-relevant information through quantification. Among other characteristics, they can measure software productivity and quality accurately and plan and estimate software projects precisely [51]. Moreover, teams use metrics for project planning and quality improvement, supporting progress tracking, identifying problems, and motivating people [62].

Over time, the focus and also the possibilities to make software development measurable have changed. In traditional (typically large) measurement programs, project management mainly sets measurable goals and tracks their achievement according to a specific plan [35]. This procedure contrasts with the agile notion of measurement, which puts the team in control of responding flexibly and quickly to customer feedback and associated trends [10, 101]. The advent of automated tools such as Jira, Azure DevOps, or VersionOne has also greatly simplified measurement and reduced manual effort. However, if the appropriate tools or data are lacking, this can lead to severe challenges in operationalizing metrics [90]. Furthermore, the absence of data quality can contribute to a false sense of security and cause inaccurate conclusions. It is fundamental to create a shared understanding for the interpretation of metrics to avoid misconceptions. On a team and cross-team level, the utilization of metrics can also spark conflicts that promote detachment from the collective vision. In addition, team comparability, wrong incentive setting, and missing transparency of organizational interrelationships may provoke pressure and demotivation [83, 91]. Therefore, it is necessary to make metrics accessible and visible to everyone and also collect metrics regularly. Metrics programs should engage teams and enable them customizing the metrics to their needs, thus building trust in the measurement [17, 91]. Furthermore, the literature recommends linking metrics to goals and aligning the available data and processes with the measurement program [91, 104]. Moreover, organizations should ideally define metrics standardized at all organizational levels and roll them out iteratively based on feedback [58, 83, 89]. Both known challenges and proven success factors have to be considered when introducing and managing metrics.

Adoption and management of metrics and strategic goals in large-scale agile software development

This thesis aims to investigate the adoption and management of metrics and strategic goals in large-scale agile software development. Hence, it serves the topic of metrics in large-scale development efforts, which is a high priority within Dingsøyr and Moe's research agenda for large-scale agile software development [27]. Prior literature focuses primarily on the adoption of metrics in the context of single-team settings. For example, Ram et al. examined success factors of metrics operationalization within two single agile teams in their case study [91]. Choras et al. validated a catalog of agile development process metrics with a Scrum team of a small-size company [18]. An exception is the exploratory case study by Korpivaara et al., in which they explored performance goals and metrics in two business units of a scaled agile organization. Despite their contribution, they argue that there is still a need for further studies to understand the interplay between metrics, strategic goals, and the scaled environment [58]. These aspects form the first part of the research gap. The second part of the research gap addresses what Oza and Korkala call a "reference framework for a metrics program." Accordingly, there is no distinct approach that introduces metrics and at the same time thinks holistically about the different goals, information needs, responsibilities, and roles of multiple software development teams. In this context, holistic also means continuously evolving the measurement program and its stakeholders [83]. Even though Ram et al. roughly outline a process for operationalizing process metrics, a scientifically grounded reference for the systematic adoption and management of metrics in large-scale agile software development is missing so far. Therefore, this thesis proposes a holistic approach for the introduction of metrics in the large-scale agile context, which at the same time provides the opportunity to professionalize the measurement program and its stakeholders. The proposed approach builds on insights from previous software measurement research and combines several proven methods and models. These include the GQM method, which guides and supports the alignment of metrics with goals and the agile development process. Boerman et al. and Mahnic and Zabkar demonstrate the practical value of a goal-based approach in their research on agile software development measurement [13, 70]. Furthermore, the proposed approach integrates the metrics team maturity model by Meding et al. [73]. Using the model, stakeholders can assess their maturity level based on various dimensions and use the obtained information to design and evolve the measurement program more efficiently. Improving the maturity level reinforces the holistic nature of the approach. In addition, the approach incorporates essential success factors for the use and adoption of metrics in large-scale agile software development.

The approach and its execution together form a comprehensive artifact. The approach derives from the findings of an analysis of related literature and the exchange with an IT-provider for insurance software in the context of a case study. The execution of the approach represents the instantiation of the artifact in two versions. The instantiation versions contribute to understanding the characteristics of agile roles and events, as well as challenges and recommendations in introducing metrics and improvement of measurement programs in large-scale agile environments.

1.2. Research objectives

First, the thesis aims to investigate the current state of the metrics in their relationship to strategic goals within the context of large-scale agile software development. The investigation of the current state addresses the aspects of the first part of the research gap. Therefore, the research question to be answered in the first step is as follows:

> **Research question 1:** How are metrics in relation to strategic goals currently being used in large-scale agile software development?

The first research question addresses the use of metrics and strategic goals in the large-scale agile context. A literature review first serves to elucidate various aspects such as the metrics and goals themselves and their agile deployment environments to answer the question. A case study follows and enriches the results from the analyzed literature. In addition to the metrics used and related strategic goals, the case study focuses on the distribution of responsibilities in the context of agile roles and events. Similar to the studies of [58], [83], [90], the research question also targets challenges in the adoption and use of metrics. Furthermore, answering the first research question sets the foundation for the second research question, which asks:

> **Research question 2:** How can different stakeholders establish and manage metrics and strategic goals to improve the current large-scale agile software development measurement program?

The second research question is mainly related to the second part of the research gap, which emphasizes improving the current situation by providing a reference framework for metrics and strategic goals [83]. The reference framework includes the establishment and management of metrics and goals involving the different levels of responsibility. In answering the second research question, the thesis ties in with the established goal-based GQM method to establish new metrics [13]. This method has proven successful for establishing agile team-level metrics [13, 70]. For the management of metrics programs and the associated stakeholders, the thesis builds on the metrics team maturity model to answer the second research question. The model helps to quantify the maturity level based on the characteristics of successful metrics teams. With the result, four participating case study companies of medium to large size in the environment of agile and large-scale agile practices were able to identify their factors to improve [73]. The thesis combines the knowledge of both methods/models and synthesizes them into one approach. In addition, the approach establishes enhancing elements that emerge from the literature review and the case study. The thesis also documents the approach and its execution. Finally, the case study participants evaluate the approach through two surveys.

1.3. Research approach

The following section provides a detailed overview of the applied methodology and research approach of this thesis.

This master's thesis consists of two established research approaches: the design-science paradigm and the case study methodology. Hevner et al. [44] introduced the design-science paradigm as one of the two dominant research paradigms in the field of Information Systems (IS). Design science aims at solving important organizational problems through building and evaluating new and innovative artifacts that address business needs. For this purpose, Hevner et al. [44] propose seven guidelines that describe requirements for conducting effective design science research in the IS discipline. Peffers et al. [86] complement the paradigm with a sequential process model that describes the flow of design science research using six activities, which derive from the guidelines of Hevner et al. and other scholars. These six activities are problem identification, objective definition, design and development, demonstration, evaluation, and communication.

Sein et al. [102] argue that the sequential process separates building from evaluating the artifact. As a result, the sequential process does not meet the needs of a research method that has built-in relevance and rigor cycles for designing innovative artifacts. Therefore, Sein et al. [102] suggest an improved action design research method that conceptualizes IT artifacts as ensembles and the activities of the research process as inseparable and inherently intertwined. The action design research method includes four stages and seven associated principles that reflect the methodological advancement's underlying assumptions, beliefs, and values. This thesis harnesses the methodological approach of the action design research method to realize the envisioned objective of developing a frame of reference for goal-based metrics adoption in large-scale agile software development. The impetus for the research of the thesis came from the case organization itself, which showed a specific demand for the introduction of new metrics in different business divisions. Hence, Stage 1 - Problem Formulation - follows the principle of practice-inspired research according to Sein et al. The thesis also pays attention to the second principle of a theory-ingrained artifact by incorporating various theoretical sources from a literature review. These sources include, for example, the GQM method, success factors in the introduction and use of metrics, or the metrics team maturity model. The specific problems in the context of this design science research are often the missing alignment between metrics and goals [47, 83], the difficulty of deriving actionable value from metrics [90], and the lack of guidance for metrics adoption [58]. In Stage 2 - Building, Intervention, and Evaluation - the problems and theoretical premises adopted in the first stage are used to create an initial design of the artifact. The initial design evolves iteratively through reciprocal shaping and concurrent evaluation in exchange with the organizational environment and its stakeholders (Principles 3 to 5). Interviews and two surveys serve as evaluation instruments at different intervals. The resulting artifact is:

- An approach for goal-based metrics introduction and improved management of measurement programs in large-scale agile software development.

The artifact emerges through organization-dominant action design research efforts. However, the interwoven nature of building the IT artifact, intervening in the organization, and evaluating is limited due to the time constraints of the master's thesis. Stage 3 - Reflection and Learning - accompanies the first two stages and helps reflect on the organization-specific design and execution of the approach. In this way, the artifact can be continuously shaped with organizational use, perspectives, and participants through the so-called guided emergence of the sixth principle [102]. The outcomes of the organization-specific design and execution of the approach translate into a generalizable solution instance in Stage 4 - Formalization of Learning. In summary, the action design science approach aims to answer the second research question.

The second part of the research approach of this thesis relies on the case study methodology, as described by Runeson and Höst [95] for software engineering. Contrary to the guideline of Hevner et al., the case study method supports the complete design science process and not only the evaluation of the artifact design. Even in the early stages of the research, the exploratory nature of the case study contributes to capturing the current use of metrics and goals in scaled agile development and formulating the associated challenges. Methodologically, this involved conducting semi-structured interviews, informal interviews, agile meeting observations, and documents throughout the case study research. These qualitative methods are intended to answer the first research question. A more detailed description of the multi-case study design can be found in Section 4.1.

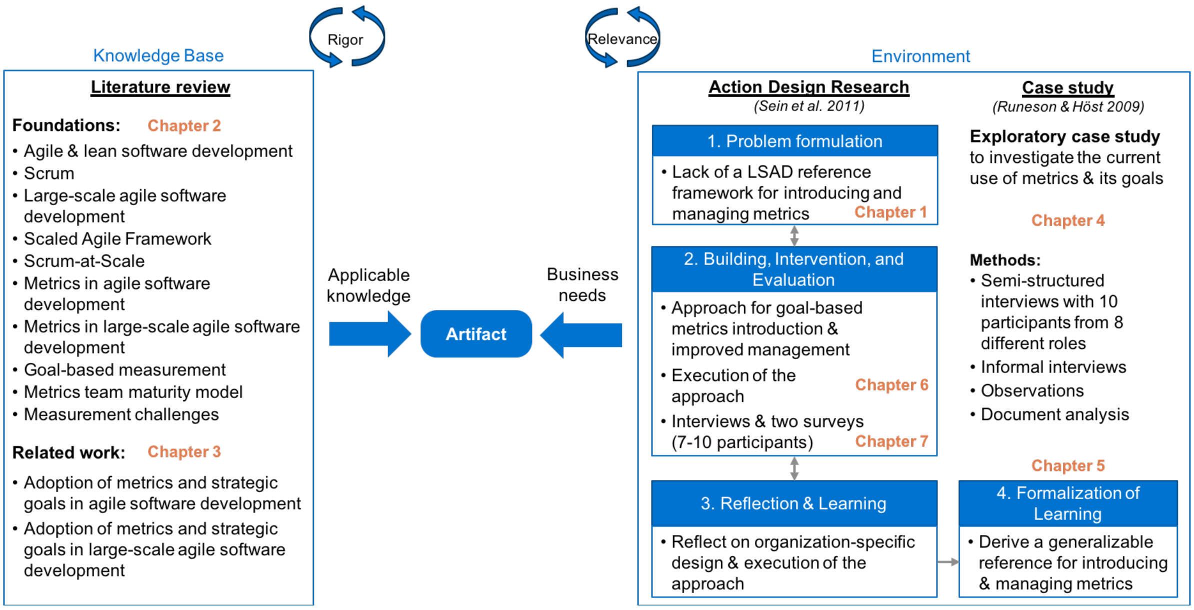

Figure 1.1.: Overview of the research approach with the mapping of thesis chapters

Figure 1.1 provides an overview of the entire research approach, the methods used with their sources, and the different activities and stages. In addition to the methodological approach, the figure shows a visual mapping of the scientific approaches and design science activities with the chapters of this thesis. The combination of the various research approaches focuses on the optimal pursuit of the research objectives, ensuring relevance and rigor of the research. Relevance stems from the environment that defines the problem space, builds a solution incorporating business needs, and evaluates them using case study and action design research. Rigor is achieved by appropriately applying existing foundations and methodologies that coalesce into a knowledge base through literature review. These two complementary and reciprocal concepts eventually result in the artifact.

The thesis structure is as follows: Chapter 1 introduces the topic of the thesis and defines the research objectives and the research approach. Chapter 2 sets the theoretical foundation and presents the relevant definitions and findings of previous research. Chapter 3 summarizes the key publications in the context of this thesis. Chapter 4 describes the case study conducted and answers the questions raised in this context. Chapter 5 presents the approach to introduce metrics and improve the measurement program management as generalization of the solution artifact within the formalization of learning stage. Chapter 6 describes the two instantiation versions as the execution of the approach. Chapter 7 presents the results of the evaluation of the solution artifact. Chapters 8 and 9 conclude the thesis by summarizing the work, identifying future work, presenting the key findings, and explaining the limitations of the thesis.

2. Foundations

This chapter describes the theoretical foundations necessary for the upcoming chapters of this thesis. The goal is to introduce relevant definitions, concepts, and findings of previous research for a common understanding of the research of this thesis. This chapter divides into three sections. Section 2.1 introduces the basics of agile and lean software development. Then, Section 2.2 discusses agile software development at a large-scale level. Finally, Section 2.3 lays the groundwork for the core of the thesis: metrics and goal-based measurement.

2.1. Agile and lean software development

This section addresses agile and lean software development in general. First, Subsection 2.1.1 presents the history of the emergence of agile software development and the further evolution of the scientific discourse on this topic. The section continues with subsections on the definition of agile software development (2.1.2) and the values and principles of agile software development (2.1.3). Subsection 2.1.4 describes the most important agile method: Scrum. The section concludes with a description of lean software development and its distinction from agile software development (Subsection 2.1.5).

2.1.1. The evolution of agile software development research

Agile software development methodologies began their evolution in the 1990s in the wake of technological advances and upheaval in the computer industry. The availability of faster PCs with larger memory capacities at affordable costs, the development of web technologies, and visual programming technologies expanded the opportunities and ambitions of software development. It was now possible to develop smaller software projects with a few developers over a short period and to implement large and complex systems that required hundreds of developers over a more extended period. The increase in complexity combined with the pressure to deliver software faster in response to changing and growing customer requirements made it more and more difficult for software developers to plan the entire system development process [49]. Besides, the processes, tools, documentation, and project management approaches that had been in place until then proved to be too complicated and increasingly left out the human side [45].

In response to these challenges, agile methodologies were then designed and continuously refined. For example, the probably best-known agile software development methodology Scrum was first published in 1995 [99, 101]. Other well-known methodologies, such as Dynamic System Development Method (DSDM) [75] and Extreme Programming (XP) [9, 11], were developed in 1994 and 1999, respectively. Early guidebooks and papers about these popular methodologies fueled the discussion on agile in the software development community and facilitated the early agile adoption in organizations [12, 68, 118].

Over time, agile methodologies became more widespread, resulting in more studies of their application. Therefore, research was initially mainly concerned with the adoption of agile methodologies and agile practices. Another research domain dealt with human and social aspects, such as collaboration in co-located or distributed agile teams [30, 46]. Many of those fundamental issues remain relevant in research and practice until today due to the ongoing transformation that most companies find themselves in [45].

In the early and mid-2000s, the research investigated synergies between agile software development and well-established software engineering concepts, like, for instance, requirements engineering [84], usability [50], and software architecture [81]. At about the same time, the classic research field of agile software development expanded to the domains of knowledge and information management, large-scale agile, and embedded systems [45].

In the late 2000s and early 2010s, respectively, two other well-established domains in software engineering, namely Capacity Maturity Model Integration [87, 113] and game design [6], were integrated into the field of agile software development research [45]. Agile software development research also spread into domains closely related, as product line engineering [43, 78] and Software development and IT operations (DevOps) [40] during the mid-2010s. Over the last few years, several publications dealt with emerging technologies, such as microservices, big data, or the Internet of Things (IoT). This trend is expected to continue and explore the potential of the not-so-new research domain of Artificial Intelligence (AI) in the context of agile software development [45].

2.1.2. Agile software development definition

The findings of Hummel's systematic literature review show that there is no generally accepted definition of agile software development [48]. Kettunen et al. confirm this statement and provide a comprehensive list of different definitions in their doctoral dissertation [54]. Most of the definitions refer only to some basic properties of certain agile methodologies, indirectly leaving out the concept of agility.

The most common definition goes back to the Agile Manifesto, which originates from many agile properties within other definitions [10]. The Agile Manifesto is a document created in 2001 at a meeting of 17 software development experts. It includes four values and twelve principles (see Table 2.1) based on the group's experience combined with existing software development practices (e.g., prototyping and continuous delivery) that should be promoted in the future [49, 63]. The novelty lies in aggregating individual principles and practices into one basic framework, as Williams and Cockburn noted [116]. Criticism of the Agile Manifesto is mainly directed at the lack of theoretical foundation and the fact mentioned above that derived agile methodologies focus too much on a few but not on all agile principles of the Manifesto [21].

In his study, Conboy develops a more research-based and comprehensive definition of agility to address the conceptual shortcomings [20]. He contrasts and compares the three terms of agility, flexibility, and leanness involving related disciplines. Finally, he summarizes agility as:

> "the continual readiness of an information systems development method to rapidly or inherently create change, proactively or reactively embrace change, and learn from change while contributing to perceived customer value (economy, quality, and simplicity), through its collective components and relationships with its environment" [20].

According to Hummel, this definition is the second most cited in the literature, followed by the definition of Abrahamsson et al. [48]. Abrahamsson and his colleagues state that the following attributes characterize agile methods: incremental (small software releases, with rapid development cycles), cooperative (a close customer and developer interaction), straightforward (the method itself is easy to learn and to modify, and it is sufficiently documented), and adaptive (the ability to make and react to last moment changes) [1, 3].

Other definitions include even more aspects that characterize agile software development, which go beyond those already mentioned. In his definition, Cockburn emphasizes light-but-sufficient rules that should be used to guide agile project work [19]. Some authors highlight the important aspect of feedback necessary for learning, as Schuh, Subramaniam and Hunt do within their definitions [98, 109]. The broadest of all definitions provide Nerur and Balijepally. Their definition incorporates other additional aspects, such as strategic thinking within uncertain and complex environments and the properties of self-organizing teams and role interchangeability originated from holographic organization theory [80].

2.1.3. Agile software development values and principles

The Manifesto for agile software development is based on four values [10]:

1. **Individuals and interactions** over processes and tools
2. **Working software** over comprehensive documentation
3. **Customer collaboration** over contract negotiation
4. **Responding to change** over following a plan

Table 2.1 contains the principles of the Agile Manifesto together with the associated emphasis (adopted from Laanti et al. [63]). These principles and their interpretations are addressed extensively in the literature [1, 15, 19, 37, 77, 115]. The first principle emphasizes four aspects indispensable for agile software development: customer satisfaction, software quality, and early and continuous deliveries. Compared to traditional software development, agile practices aim to satisfy customers by making valuable parts of the final product accessible early in the development life-cycle. The goal is to develop and test software quickly and incrementally in shorter time intervals. This goes so far as to accept changing requirements late in the development, as expressed in principle 2. At the same time, software should be delivered at regular, preferably short, intervals (principle 3). Principles 4, 5, and 6 emphasize direct communication and trusting collaboration in a working environment that motivates developers. The authors of the Manifesto recommend paying special attention to the completion of the parts of the product that lead to the desired final product - called working software - to increase customer satisfaction (principle 7). Another important aspect reflected in principle 8 is the long-term and sustainable development that customers, developers, and sponsors constantly maintain. The Agile philosophy stresses the importance of software quality and excellent design in principle 9. The tenth principle is expressed in the simplicity of development and the ability to avoid unnecessary effort. Agile principles 11 and 12 also encourage development in self-organized teams that are constantly willing to learn and adjust their behavior accordingly to increase effectiveness and develop the best architectures, requirements, and designs [77].

Bustard et al., Williams' industry surveys show that Principles 1, 3, 5, 7, 12 of the Agile Manifesto have strong support among practitioners. Principle 11, however, had the lowest approval rating in both surveys. For principles 2 and 6, the two studies yielded different results. Principle 2 had a higher ranking in Williams' survey than in Bustard et al.'s survey. For Principle 6, it is the other way around. Based on the two studies, it can be seen that the level of acceptance in practice varies within the twelve agile principles.

Dingsøyr et al. literature analysis divides the spectrum of perspectives of agile software development methodologies into three areas depending on the degree of emphasis on the values and principles of the Agile Manifesto. The studies and organizations examined fall on a continuum that ranges from strict adherence to loose recommendations. According to this, Avid Agile followers try to pursue the values and principles of the Manifesto strictly and only deviate from them should the integrity of the original idea be compromised. The Pragmatic Agile perspective takes a much milder approach to methodological compliance and allows flexibility to adapt the methodology to the situation. In between is the Inclusive Agile view, which adopts the agile values and principles but also makes environmental changes and combines different methods if necessary [118]. Overall, it can be stated that there is the possibility to benefit from the agile principles even if they are only partially implemented [64].

2.1.4. Scrum

Scrum is the most prominent agile methodology. The "14th State of Agile Survey" from 2020 reveals that 56% of the companies use sole Scrum and 76% when adding combinations with other methodologies [106]. Scrum was first presented in 1995 at the OOPSLA conference by Ken Schwaber and Jeff Sutherland [99]. The idea was to provide people, teams, and organizations with a lightweight framework that would result in adaptive solutions to complex problems and, thus, create value. To do this, it leverages the theories of empiricism as an experience-driven decision-making foundation and lean thinking introduced in Subsection 2.1.5. Scrum follows an iterative, incremental approach, which manifests in the event of the *Sprint*. The Sprint implements the cornerstones of empirical process control: *transparency, inspection, and adaptation*. The transparency cornerstone requires that the process and the work be visible to those doing the work and those receiving the work. This visibility enables

No.	Agile software development principle	Emphasis
1	Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.	Customer satisfaction, continuous delivery, software quality, early deliveries
2	Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.	Adaptability, responsiveness, customer orientation
3	Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.	Frequent deliveries
4	Business people and developers must work together daily throughout the project.	Collaboration
5	Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.	Motivated employees, Good work environment, support, trust
6	The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.	Direct communication
7	Working software is the primary measure of progress.	Progress measurement through working software
8	Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.	Sustainable development, long-term partnership
9	Continuous attention to technical excellence and good design enhances agility.	Software quality, design focus
10	Simplicity - the art of maximizing the amount of work not done - is essential.	Simplicity, continuous improvement
11	The best architectures, requirements, and designs emerge from self-organizing teams.	Self-organizing teams
12	At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.	Continuous improvement and learning, effectiveness

Table 2.1.: Principles of the Agile Manifesto [63]

inspection, which would be misleading and wasteful without transparency. Frequent and careful inspection of Scrum artifacts and goal progress ensures that potentially unwanted deviations or problems are discovered. If the inspection reveals deviations in the process or dissatisfaction with the product, they must be corrected and adapted immediately. Inspection is hence a prerequisite for adaption as well [101].

The basic unit of Scrum is the *Scrum Teams*, which according to the Scrum Guide, should consist of less than ten people [101]. J. R. Hackman and R. J. Hackman's research locates the optimal team size at four to five members. In general, smaller teams are considered to promote efficient communication and more productive working [41]. Moreover, the successful use of Scrum depends on the extent to which the team members live the following five values: *commitment*, *focus*, *openness*, *respect*, and *courage*. "These values give direction to the Scrum Teams concerning their work, actions, and behavior", as Schwaber and Sutherland note. These teams consists of a *Scrum Master (SM)*, a *Product Owner (PO)*, and developers. They cover all the necessary skills to complete the Sprint in a value-added manner. They are jointly responsible for the success and also manage themselves. Developer responsibility lies in compiling the *Sprint Backlog*, adhering to the *Definition of Done*, daily alignment of the plan towards the *Sprint Goal*, and mutual accountability checks. The PO covers the part of the team's responsibility that relates to the value of the product. Therefore, the PO develops and communicates the *Product Goal*, creates, organizes, and communicates the *Product Backlog* items, and ensures that the items are transparent, visible, and understood. In contrast, the SM is responsible for the process of Scrum and, consequently, the effectiveness of the team. The scope of responsibilities includes coaching the Scrum Teams on self-management, creating valuable and Definition of Done compliant increments, adhering to and constructively executing Scrum events, and initiating the removal of impediments. In addition, the SM supports the PO in effectively defining Product Goals, managing the Product Backlog, communicating it clearly to the team, and establishing empirical product planning. Overall, the SM serves the organization in leading, training, and coaching the adoption of Scrum, planning and advising on Scrum implementations, and supporting employees and stakeholders in removing impediments [101].

The Sprint as an event encapsulates in Scrum all four events: *Sprint Planning, Daily Scrums, Sprint Review*, and the *Sprint Retrospective*. Within recurring fixed periods of one month or less, the Scrum Team strives to get closer to the Product Goal by creating a Product Increment. At the beginning of each Sprint, the Scrum Team decides collaboratively in the Sprint Planning event what the Sprint Goal is, which Product Backlog items should be part of the Sprint Backlog, and how they will be achieved. The 15-minute Daily Scrum event inspects the Sprint Goal progress and enables the Sprint Backlog to be adjusted or impediments to be named. The purpose of the Sprint Review is to present the outcome of a Sprint to the key stakeholders so that they can inspect the Product Increment and determine future adaptations in line with the Product Goal. In the Sprint Retrospective, the Scrum Team inspects the completed Sprint regarding people, interactions, processes, tools, and their Definition of Done and plans steps on how to improve the quality and effectiveness [101].

Scrum artifacts act as a transparency mechanism for gaining information about the team's

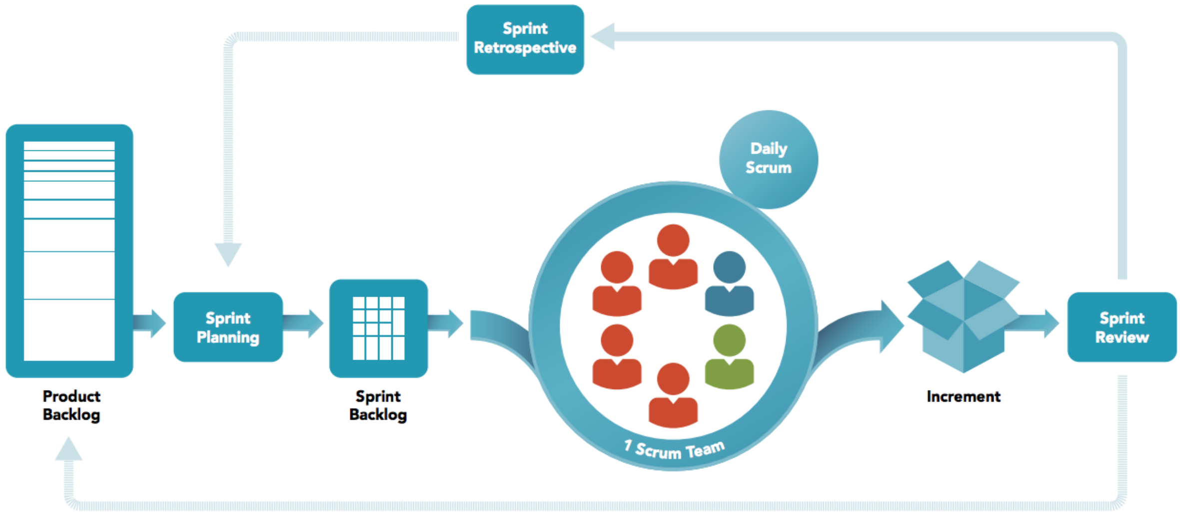

Figure 2.1.: The Scrum Framework [100]

work and the product's value. Each of the artifacts contains a commitment that measures progress. The Product Backlog is an ordered list of product improvements that align with the Product Goal. The Sprint Backlog refers to the Sprint Goal and contains only those Product Backlog items that the Scrum Team selects for the Sprint. The Product Increment is a concrete sub-step towards the Product Goal and increases the progress compared to previous increments. The Product Increment is measured based on the Definition of Done, which formally describes its status [101]. An overview of the Scrum Framework with its events and artifacts depicts Figure 2.1.

2.1.5. Lean software development

As most of the literature notes, the concept of leanness goes back to Toyota's manufacturing system [88, 114]. The idea was to reduce costs by eliminating waste, in other words, doing more with less. Just-in-time production, minimum reasonable inventory, and the production flow control method *KANBAN* (origin of the kanban board used in software development) embody this principle of economy. The zero-defect policy reflects another aspect of leanness, which is about quality from the customer's point of view. In addition, lean organizations rely on simple structures that enable simple measurements of quality and value [20]. Conboy summarizes these aspects as follows:

> "leanness is the contribution to perceived customer value through economy, quality, and simplicity" [20].

Wang et al. state that lean software development consists of three elements: lean concepts, lean principles, and lean practices. The lean concepts echo parts of [20]'s definition and distinguish five lean guiding concepts. First, *value* is defined by the customer, and it is important to have a clear understanding of it. Second, a map makes the *value stream* and each associated step in the process visible and categorizes the added value. The third concept focuses on the continuity of the production process *flow*. The fourth concept relies on customer orders to *pull* the product to ensure that nothing is built before it is needed. With the fifth concept, teams pursue perfection in the process by continuously identifying and removing waste [114]. According to M. Poppendieck and T. Poppendieck, lean principles include: eliminate waste, build quality, create knowledge, defer commitment, deliver fast, respect people, optimize the whole [88]. Lean software development practices embrace addressing bottlenecks, where the cumulative flow diagram is applied. Added to this are lean practices such as avoiding too much local optimization, continuous improvement (Kaizen) and relentless self-reflection, transparency of project status and visualization of all work elements, measurement and management, and workflow analysis and design. Overall, it is evident that lean concepts, principles, and practices have found their way into agile software development [114].

However, the concepts of lean and agile differ in three main ways. First, being agile requires responding well to variability, while leanness works well in controlled environments. Second, organizations are lean when they are cost-efficient and productive, whereas agile ones learn fast to become cost-efficient and productive even though they are not right from the beginning. Third, ultimate leanness eliminates all waste; by contrast, agile requires eliminating waste, but only to the extent that the ability to respond to change is not constrained [20].

2.2. Large-scale agile software development

This section covers the theory of the research environment, namely large-scale software development, and all the main aspects that belong to it. The first Subsection (2.2.1) presents different definitions and interpretations of large-scale software development. Subsection 2.2.2 briefly introduces some of the existing frameworks. Finally, Subsections 2.2.3 and 2.2.4 describe the relevant case study frameworks SAFe and Scrum-at-Scale in more detail.

2.2.1. Large-scale agile software development definition

During the last two decades, agile methodologies have become the dominant software development mode [45]. Initially, it was assumed that agile values and principles were best applied in the context of small, co-located teams with on-demand access to developers. Smaller agile teams were originally sized to up to 50 people [2, 116]. However, this classic understanding has changed over time, with agile methodologies being used in new environments such as global, distributed teams [30, 46] or large development organizations [92]. At the same time, this approach is criticized for its insufficient focus on architectural decisions and the shortcomings of classical agile methods in large-scale software development [26, 92]. The difficulty in adopting agile practices increases when they need to be scaled. *Scalability* in software engineering is defined as "the property of reducing or increasing the scope of methods, processes, and management according to the problem size" [65].

The term "*large-scale agile software development*" has different interpretations in the literature depending on the factors considered and their context. These factors include project costs, project duration, number of requirements, number of people involved in the development, lines of code in the solution, number of development sites, or number of teams [26, 27, 93]. Many of these factors prove insufficient, which is why Dingsøyr et al. advocate using the number of teams to define large-scale agile. According to this, agile software development projects with two to nine teams are considered large-scale, where coordination should occur in a forum that is superior to the teams. If more than nine teams are involved, the authors refer to this as very large-scale software development. Here, even several of these forums are necessary. This definition excludes such arrangements that apply agile methods in a large organization because this represents a separate research direction called enterprise agile [26, 28]. Two general interpretations of large-scale agile can be derived from this:

1. the application of agile methodologies in multi-team settings
2. the use of agile approaches in large enterprises

In addition, two other views of large-scale complete the collection of possible interpretations:

3. the usage of agile methodologies in large teams
4. the company-wide deployment of agile approaches

In general, regardless of the perspective, large-scale agile software development relies on many people and many systems, which have many interdependencies and, thus, increase complexity [93]. In the further course of the thesis, the first and the fourth interpretation will be used.

2.2.2. Frameworks in large-scale agile software development

Scrum, XP, DSDM, and Kanban are the most common agile methodologies that organizations use when migrating to an agile approach [45]. These methodologies were developed for the application at the team level and leverage their strengths best at this scale. However, the increasing adoption across multiple teams and even organization-wide at different locations made it necessary to restructure teams' alignment, collaboration, and interdependence within the agile setting. This structural change also required a methodological representation. In response to this, so-called large-scale agile frameworks have been developed, which are based on agile principles, combine various aspects of the methodologies mentioned above, and integrate them into the life-cycle of project and portfolio management [4]. Over the last two decades, several frameworks have been proposed. Among them are Large-Scale Scrum (LeSS), SAFe, Discipline Agile Delivery (DAD), Spotify, Recipes for Agile Governance in the Enterprise (RAGE), Scrum-at-Scale, and Nexus. The key data for each framework can be found in an overview in Table 2.2 (adapted from [4, 32]). The frameworks SAFe and Scrum-at-Scale are described in the following subsections 2.2.3 and 2.2.4.

LeSS	
Methods & practices adopted	Scrum was fully adopted including additional practices for large projects
Team size	Up to ten Scrum teams (of seven people) or up to a few thousand people
Organization type	Large traditional organization and enterprises
DAD	
Methods & practices adopted	Kanban practices, Scrum, Agile Modeling, the Unified Process, XP, Test Driven Development and Agile Data
Team size	200 people or more. It also supports small and medium teams.
Organization type	Multiple organizations and enterprises
Spotify	
Methods & practices adopted	Allow Kanban, Scrum, DevOps and Lean Startup
Team size	250 to 300 people with around 30 teams
Organization type	Enterprises specifically similar to Spotify
RAGE	
Methods & practices adopted	Allow Scrum, Kanban, plan-driven development and hybrid approaches
Team size	Different sizes supported
Organization type	Traditional and agile enterprises
Nexus	
Methods & practices adopted	Scrum with additional practices in solving the dependency related issues in multiple teams
Team size	Three to nine Scrum teams
Organization type	Portfolio level for medium sized projects

Table 2.2.: Overview of large-scale agile frameworks [4, 32]

2.2.3. Scaled Agile Framework

With a market share of 35 percent, the Scaled Agile Framework (SAFe) is the most widely adopted and well-researched agile scaling method [106]. Dean Leffingwell, one of the co-authors of the Agile Manifesto, established SAFe in 2011 together with Drew Jemilo to help companies deliver new products in the shortest sustainable lead times with the best possible quality and value. SAFe also synchronizes alignment, collaboration, and delivery for multiple agile teams. The framework supports the full range of scaled agile development environments covered by four different configurations: Essential SAFe, Large Solution SAFe, Portfolio SAFe, and Full SAFe. The *Essential SAFe* configuration forms the core of SAFe and lays the foundation for all other SAFe configurations built on top of it. The main objective of *Portfolio SAFe* is to align portfolio execution with business strategy by organizing agile development around the flow of value through one or more value streams. The *Large Solution*

SAFe configuration is suitable for developing large and complex solutions, which typically require multiple ARTs but do not mandate portfolio operations. *Full SAFe* provides the most comprehensive version of the framework. It helps organizations with several hundred employees build and maintain large integrated solutions that include all levels of SAFe: Team, Program, Large Solution, and Portfolio [66]. The following paragraphs describe the core elements of SAFe using the Essential SAFe configuration, as shown in Figure 2.2.

The SAFe foundation consists regardless of the particular configuration, by definition, of the elements: lean-agile leaders, core values, lean-agile mindset, SAFe principles, SAFe implementation roadmap, and SAFe Program Consultants (SPC)s (gray shaded area in the lower part of Figure 2.2). The first element of the SAFe foundation intends to turn managers into lean-agile leaders and train them in leaner ways of thinking and working. Second, the SAFe foundation includes four core values that describe the ideals and beliefs essential to applying the framework. The first value is alignment, which is respected when every team member and ART understands the strategy and their role in implementing it and works toward the same goals. The second value of built-in quality ensures that customer satisfaction is the top priority and that value delivery moves quickly and more predictable. The third value is transparency, which builds trust and is a prerequisite for performance, innovation, risk-taking, and continuous improvement. The fourth and final value manifests itself in the innovation, speed, and agility of the stable Agile Release Trains and is called program execution. As the third element of the SAFe foundation, the lean-agile mindset combines beliefs, assumptions, and actions of SAFe leaders and practitioners based on the Agile Manifesto. The foundation also includes the nine SAFe principles: (1) take an economic view, (2) apply systems thinking, (3) assume variability; preserve options, (4) build incrementally with fast, integrated learning cycles, (5) base milestones on an objective evaluation of working systems, (6) visualize and limit work-in-progress, reduce batch sizes, and manage queue lengths, (7) apply cadence, synchronize with cross-domain planning, (8) unlock the intrinsic motivation of knowledge workers, and (9) decentralize decision-making. The SAFe implementation roadmap guides organizations on implementing the changes necessary to become a lean enterprise. The final element of SAFe foundations is SPCs, who are change agents who supply their technical knowledge of SAFe with intrinsic motivation to improve software and system development processes [66].

In Essential SAFe, there is the team level and the program level, which together form the organizational structure of the Agile Release Train. The two levels are closely linked and follow agile and lean practices. The team level involves roles, artifacts, events, and processes comparable to those in Scrum. All teams within the ART develop iteratively in fixed-length time-boxes that share the same duration and have identical start and end dates. Each iteration ends with the completion of a valuable increment that adds new functionality to the existing solution. Multiple iterations combine into a so-called Product Increment (PI) that aggregates larger, system-wide functionality across teams and delivers it on demand. SAFe teams have a choice of ScrumXP, Team Kanban, or a hybrid model as a process or method for self-organizing cross-functional teams of five to nine people. The former is a merger of the Scrum framework for project management and XP-derived software engineering practices, while Team Kanban helps teams facilitate value flow by visualizing workflows and measuring throughput. A ScrumXP or Kanban Team consists of a Scrum Master, a Product Owner, and the development team with developers, testers, and other specialists. In addition to events almost identical to Scrum, such as Iteration Planning, Iteration Review, Iteration Execution, Iteration Retrospective, and Backlog Refinement, SAFe also offers the *Innovation and Planning Iteration* event. The Innovation and Planning Iteration occurs at each PI, serves as an estimation buffer for meeting PI objectives, and provides dedicated time for innovation, PI planning, and continuous learning [66].

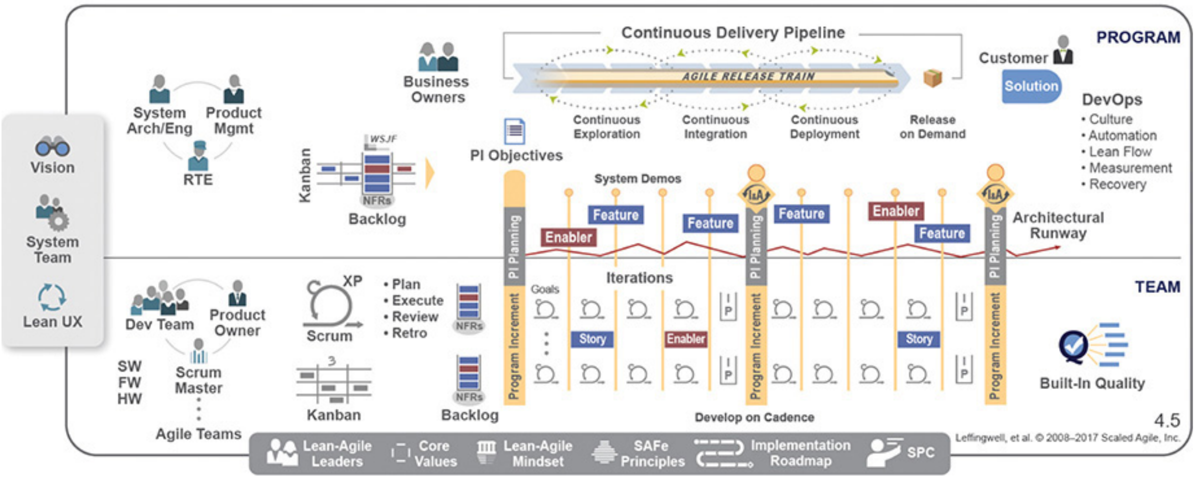

Figure 2.2.: Essential SAFe [66]

The program level is significantly reflected in the Agile Release Train, which acts as a virtual organization that bridges functional and team boundaries and eliminates unnecessary hand-offs and steps. Each ART consists of five to ten agile teams with a standard size of between 50 and 125 people. The ART operates within a PI that is typically eight to twelve weeks long. The time-box is divided, for example, into four development iterations followed by an innovation and planning iteration. During this time, each ART builds and maintains a so-called Continuous Delivery Pipeline that provides a continuous release of value to the end-user. The pipeline refers to workflows, activities, and automation and consists of the four elements: Continuous Exploration (CE), Continuous Integration (CI), Continuous Deployment (CD), and Release on Demand. Moreover, by implementing DevOps, SAFe aims to enable each ART to break down silos, thus continuously delivering new features to end-users. DevOps is a mindset and culture that entails technical practices and bundles communication, integration, automation, and close collaboration among all the people needed to plan, develop, test, deploy, release, and maintain a solution. SAFe recommends a set of augmenting roles at the program level to coordinate ARTs, align them with a shared mission, and provide the necessary lean governance. These roles include the *System Architect/Engineer*, who, as an individual or as a small cross-disciplinary team, defines the overall architecture for the system, helps in defining the non-functional requirements, or determines the design of system components and their interfaces. *Product management* represents the customer's point of view. For this purpose, it works directly with the customer but also with the product owners. It is responsible for the program backlog. The *Release Train Engineer (RTE)* takes a central role in the ART and acts as the leading Scrum Master and Coach for the train. The RTE moderates ART events and processes and supports the teams in delivering value. To do this, the RTE communicates with stakeholders to escalate obstacles, manages risk, and relentlessly drives improvement. *Business Owners* are responsible for the usability, governance, and return on investment of a solution developed by the ART from both a business and technical perspective. They are the ART's primary stakeholders and, to that end, actively participate in the following three program-level events: PI Planning, System Demo, and Inspect and Adapt. *PI Planning* usually lasts for two days and occurs periodically as part of the innovation and planning interactions. The event is ideally held in person with all ART members, hosted by the RTE. The goal is to align all teams in the ART towards a shared mission and vision. The *System Demo* provides sponsors, customers, and stakeholders with an objective overview of the new features delivered by all teams in the ART over the last iteration. For the teams, it is an opportunity to get direct feedback and get the value of their progress at the end of each iteration assessed. The velocity serves as the central measurement tool here. The *Inspect and Adapt* event is carried out at the end of each PI as a workshop with all stakeholders of the program. The agile teams demonstrate the current state of the solution and receive an assessment from the participants, followed by a structured problem analysis that results in improvement backlog items [66].

SAFe includes multiple artifacts at both the team and program levels. The primary artifact for defining functionality is *stories*. They are described in the user's language and are sized to be completed in one iteration. There are also specific story types such as the *user story*, which addresses the end-user directly, or the *enabler story*, which enables supporting activities related to exploration, architecture, infrastructure, and compliance. Agile teams create user and enabler stories mainly during PI planning or backlog refinement meetings and enter them into the *team backlog*. At the program level, *features* summarize stakeholder requirements via systems or services and provide them with a name, a benefit hypothesis, and acceptance criteria. Features are sized to fit within a PI. The ART stores the features in the *program backlog*. Furthermore, there are *program epics* that act as containers for an entire solution development initiative. The *architectural runway* as another artifact describes, among other things, existing code, components, and technical infrastructure required to support the implementation of prioritized near-term features. The ART pursues program-level *PI objectives*, which every single team breaks down into their *team PI objectives* and then refines again into *iteration objectives*. Similar to the team level, the *program kanban* method also manages and visualizes the flow of features and enablers through the Continuous Delivery Pipeline at the program level [66].

The spanning palette in the context of Essential SAFe includes at least the elements of vision, system team, and lean user experience (see the left edge in Figure 2.2), although companies are free to apply additional elements to Essential SAFe. Vision imagines the future of the software solution, incorporating customer needs and available resources. The system team supports the organization in building and using the agile development environment and applying continuous delivery pipeline practices. In SAFe, lean user experience refers to the iterative and hypothesis-driven measurement and learning loops in product development that improve user experience design [66]. Subsection 2.3.4 explains the additional spanning palette element of metrics in detail.

2.2.4. Scrum-at-Scale Framework

In practice, it became apparent that as the number of Scrum Teams within an organization grew, two major issues arose. First, the quality, quantity, and velocity of working products per team began to decline. Cross-team dependencies, duplication of work, and additional effort due to unstructured communication mainly triggered these issues. Second, the existing management structure turned out to be ineffective in achieving business agility. These issues were seen as competing priorities and the inability to relocate teams quickly to meet the market dynamics. Therefore, the Scrum-at-Scale framework was developed, which addresses the issues above through linear scalability and business agility. On the one hand, as the delivery of work products increases, the number of teams also increases to the same extent. On the other hand, the existing functionality of the single Scrum Teams is enhanced by a structure of a cross-team network, which works with a so-called Minimum Viable Bureaucracy (MVB). MVB implies the fewest number of governing bodies and processes necessary to perform the organizational functions without impeding the delivery of customer value [110].

The framework builds on the value-driven culture of Scrum, which includes the two pillars of empirical process control and the Scrum values. Transparency, inspection, and adaptation are the cornerstones of empirical process control. Also, there are the five Scrum values openness, courage, focus, respect, and commitment [101]. Overall, the Scrum-at-Scale framework helps organizations promote a positive team learning environment for working at a sustainable pace while putting customer value delivery as the highest priority [110].

In a scaled environment, the grouping of teams is optimally done based on the coordination needs for the full integration of an increment in the form of a Scrum of Scrums (SoS). For this purpose, a functioning Scrum Team with a healthy way of working can serve as a prototype. SoS is a larger team consisting of multiple Scrum Teams, each executing the process Scrum envisions for a single team. The number of members per Scrum Team remains the same. The responsibility for the fully integrated set of potentially deliverable product increments now rests with the entire SoS team. Just as in single team Scrum, Scrum-at-Scale separates process responsibility (the "how") from product responsibility (the "what"). These different areas of responsibility are reflected in the Scrum Master Cycle and the Product Owner Cycle, as shown in Figure 2.3 [110].

Both cycles of SoS require scaling the roles and events. One of the scaled roles is that of the Scrum Master, called Scrum of Scrums Master (SoSM) in the Scrum of Scrums environment. This person is responsible for the scaled events, the prioritization, and removal of impediments, the release and deployment of the cross-team efforts, and the continuous improvement of the effectiveness of the SoS. The SoSM operates in the Scrum Master Cycle ("how") and enables cross-team coordination and communication by understanding its dependencies. Responsibilities include providing all relevant stakeholders with information on product

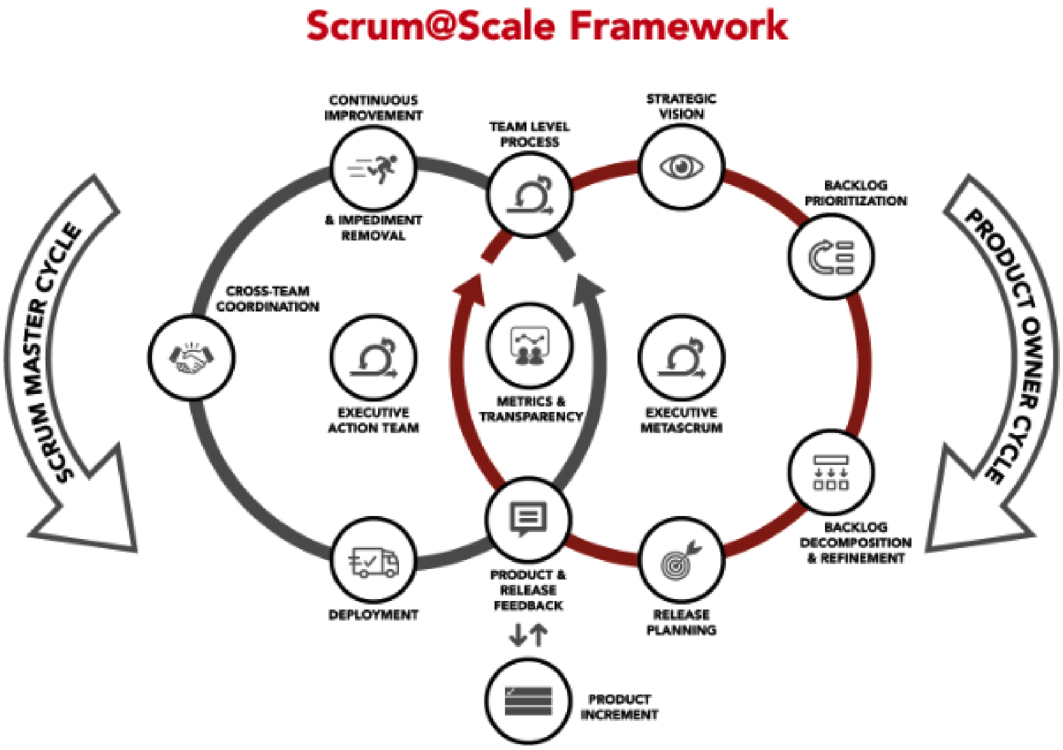

Figure 2.3.: Scrum-at-Scale Framework [110]

development progress, impediment removal status, and other metrics. The Scrum Master Cycle at the SoS level mandates holding scaled versions of the Daily Scrum and the Sprint Retrospective. The Scaled Daily Scrum event intends to foster a shared understanding among participating teams of progress and impediments. Therefore, each Scrum Team sends at least one representative to establish the communication path to the scaled level. The Scaled Retrospective brings together Scrum Masters from all teams to discuss experiments related to continuous improvement [110].

The scaling of the Product Owner Cycle is reflected in the Product Owner Team, which is led by the Chief Product Owner (CPO). At the SoS level, the teams share a common backlog created and prioritized by the CPO. At the team level, each team PO is in charge of the composition and prioritization of the team Sprint Backlog. The Product Owner Team works with the SoSM to find a harmonized *Definition of Done* for all teams. Besides, the PO Team is responsible for creating a coordinated roadmap and release plan. Another task of the team is to monitor metrics that provide insights into the product and the market. The CPO defines these metrics. Furthermore, this person defines the strategic direction of the product and organizes the MetaScrum event. This event is conceived as a forum for executives and key decision-makers to negotiate priorities, change budgets, and realign teams to achieve maximum value. SoS also comprises two events in the Product Owner Cycle, of which there are scaled versions. One of them is the Sprint Review conducted with the PO team members and chaired by the CPO. The other is the Sprint Planning with the Product Owner Team and the Scrum Masters [110].

As figure 2.3 shows, the two cycles intersect at the team level process and again at the customer feedback for the product and the release. The POs utilize the feedback to continuously improve the product and realign the Product Backlog(s) accordingly. In the Scrum Master Cycle, Release Feedback is used to adjust delivery mechanisms. In both cases, the goal is to understand the customer better and transform new ideas into new functionality. In addition, the intersection of both cycles focuses on using a minimum set of metrics to promote transparency. Achieving radical transparency is essential to maintain the ability to understand progress, processes, and products more accurately. After all, this allows data-driven decisions, reduced decision latency, and streamlined work of teams, stakeholders, or leadership [110].

The framework is complemented by two leadership groups that are part of the MVB. First, the Executive Action Team (EAT) acts as a hub within the Scrum Master Cycle and takes care of faster and optimized workflows on a higher level. This leadership team formulates guidelines and procedures that govern the collaboration between the Scrum-at-Scale world and the non-agile parts of the organization. Besides, the EAT coordinates several SoS and ensures, for example, that impediments get removed that could not previously be resolved due to the lack of financial and political resources. The EAT's activities and responsibilities result in the creation of an agile operating system. For this purpose, it oversees a Product Backlog that stems from various initiatives that promote the organizational transformation toward more business agility. These include, for example, process improvements in the form of impediment removal and standardization. All actors of the Scrum Master Cycle, i.e., SM, SoSM, and the EAT, work together to deliver the components of continuous improvement and impediment removal, cross-team coordination, and continuous customer-centric delivery (see Figure 2.3). Ultimately, however, the EAT is responsible for the quality of Scrum within the organization, and hence all Scrum Masters report to them [110].

The second leadership group is the Executive MetaScrum (EMS). This dynamic group consists of the CPO and the PO of each Scrum Team. They set the organizational vision and the strategic priorities along which all teams align. Therefore, the EMS meets at least once per Sprint and synchronizes the backlog for the entire SoS. Within the Product Owner Cycle, the collaboration of the PO Team results in a strategic vision that attracts both employees and customers, a decomposed, prioritized, and refined backlog, and a well-communicated, long-term release planning [110].

As shown in Figure 2.4, it is possible to scale multiple Scrum of Scrums if the size and complexity of the product are appropriate. The roles, artifacts, and events are expanded linearly so that, for example, a Scrum of Scrum of Scrums (SoSoS) Master coordinates the Scrum Master Cycle and multiple MetaScrums prioritize their strategic backlog at an EMS [110].

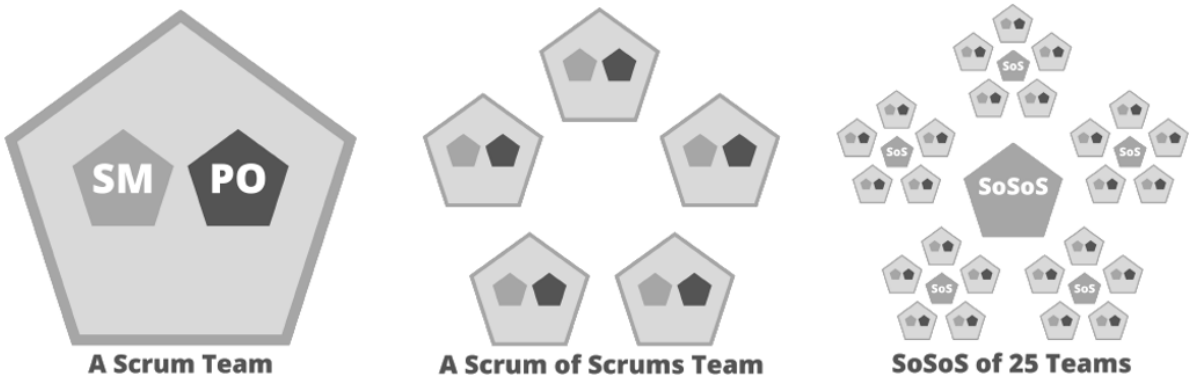

Figure 2.4.: Scaling levels of Scrum-at-Scale [110]

2.3. Metrics and goal-based measurement

The next subsections deal with the basics of the core topic of this thesis: metrics and goal-based measurement. First, Subsection 2.3.1 differentiates the term metrics from Key Performance Indicators and explains their role in traditional and agile reporting. Subsection 2.3.2 continues with the distinction between quantitative and qualitative reporting and the associated scientific findings, particularly in agile portfolio management. Examples of metrics and their classification according to different systems are presented in subsection 2.3.3. Subsection 2.3.4 then outlines metrics in the large-scale environment and addresses metrics from the SAFe and Scrum-at-Scale frameworks. Subsection 2.3.5 then introduces different approaches and methods for goal-based measurement. In particular, the Goal Question Metric method is described in more detail. Subsection 2.3.6 contains the theoretical foundations of the MeTeaM model that was applied in the action design research. Finally, the challenges that can arise in connection with measurement in the large-scale context are briefly delineated in Subsection 2.3.7.

2.3.1. KPIs and metrics in reporting

Reporting is one of the main areas of traditional portfolio management and is positively associated with its performance. The availability and sharing of information for better decision-making justify this relationship. Reporting is also considered a control structure or governance practice [79]. In comparison, reporting in the agile environment focuses less on a specific plan than on the customer value and progress. Therefore, particular emphasis is placed on communication and cooperation with external stakeholders, work task and resource transparency, and regular feedback [76, 107].

Key Performance Indicators (KPI)s and metrics are recognized as essential instruments for general reporting and are mostly presented via dashboards, scorecards, and reports. The terms Key Performance Indicator (KPI) and metric are often used interchangeably, even though they are different. The term metric is generic, while a KPI is more narrowly defined [53]. According to Eckerson:

> "A KPI is a metric measuring how well the organization or an individual performs an operational, tactical, or strategic activity that is critical for the current and future success of the organization" [31].

It turns out that KPIs focus on future outcomes that key information stakeholders need for decision making, while metrics are more geared toward aspects of the past, such as work package completion and attainment of performance goals. Both KPIs and metrics are not about fully predicting future project success or the actions required to achieve it, but KPIs can provide information about what might happen in the future if trends continue [53]. Parmenter divides metrics in terms of their criticality and time orientation into three categories. First, KPIs serve as indicators of critical performance to dramatically increase performance or goal achievement. The second category of metrics is Performance Indicators, which indicate what needs to be done to increase or meet performance. As the third metric category, Result Indicator (RI)s measure what has been achieved [85]. Many companies use metrics in an unbalanced proportion, with a tendency to declare every metric a KPI. Parmenter, therefore, recommends that ten percent of the implemented metrics should be RIs, 80 should be Performance Indicators, and ten should be KPIs [85]. Ultimately, which metric is a KPI and which is not depends on the project environment and should be evaluated individually.

In the literature, when it comes to defining the characteristics of KPIs and metrics in reporting, the *SMART* rule is often described. SMART stands for **S**pecific, **M**easurable, **A**ttainable, **R**elevant, and **T**ime-based. This rule originates from the objective setting in project management and has been adapted to identify metrics and KPIs. Other basic characteristics include purpose orientation, usefulness, target focus, accuracy, project status transparency, proactive management support, assessability of project success, and stakeholder acceptability for decision making [53].

In traditional project management reporting, there are four broad categories of metrics. The first category comprises the classic *business-based or financial metrics*, including profitability, added efficiency, and cost reduction. The second category of *success-based metrics* includes stakeholder satisfaction and milestone achievement. The *project-based metrics* encompass very traditional metrics such as cost, time, and quality. The fourth category of *project management metrics* measures, for example, continuous improvement and estimation accuracy [53]. According to Krebs exist three types of metrics for agile project reporting: *progress metrics*, *quality*, and *team morale* [60]. Stettina and Schoemaker provide a similar but more refined classification for agile reporting. In their cross-case analysis, metrics are assigned to five types of metrics [108]:

1. Performance metrics measure the efficiency of work
2. Quality metrics measure the quality of work
3. Progress metrics measure how far along you are with the work
4. Status metrics measure the current state of work

5. Contextual metrics provide measurements and information on the work

Subsection 2.3.3 applies each type of Krebs's and Stettina and Schoemaker's classification to the most important agile software development metrics.

2.3.2. Qualitative and quantitative reporting

Metrics have the function of determining the value of a project. When identifying the appropriate metrics for project management reporting, it becomes apparent that the value is not always easy to measure. Such metrics that can quickly determine the value are referred to as tangible value metrics. Tangible values are usually expressed quantitatively. These metrics often build on assumptions that may not be transparent to decision-makers and only approximately measure the actual value. Intangible metrics, however, identify the hard-to-measure value of a project and may also be captured qualitatively. There exist the issues of the subjectivity of the person measuring and the limited availability of measurement models. Despite the potential issues, intangible metrics are increasingly applied, especially in IT projects when customer or employee satisfaction is measured [53]. Kerzner assumes that a compromise between qualitative and quantitative value assessment yields the highest benefit [53]. Stettina and Schoemaker come to a similar conclusion in the context of agile portfolio management. In the case organizations of their study, a mixture of qualitative and quantitative reporting proved to be particularly effective. Qualitative reporting emerged as a technique to explore opportunities, share knowledge, provide context, and provide strategic insights. On the contrary, quantitative reporting was found to quantify initiatives and their progress, verify goals, validate value, and provide quick tactical insights. The study also makes evident that the higher the agile maturity of an organization, the higher the level of quantification of goals and initiatives. Consequently, organizations with a relatively high maturity level tend to use qualitative reporting more often at a higher organizational level and define it more loosely [108].

Similar to the distribution of roles in Scrum, Stettina and Schoemaker describe three different domains of knowledge responsibilities for reporting in agile portfolio management. First, development teams and the software architects exercise *development responsibility*. Secondly, there is the *process responsibility* that Scrum Masters and Release Train Engineers perform. The third domain is the *product responsibility* of Product Owners and Product Management. The authors assign specific types of knowledge boundaries to these responsibilities and give examples of effective boundary objects in qualitative and quantitative reporting. Table 2.3 provides an overview of qualitative and quantitative reporting and its boundary object examples. The underlying concept of knowledge boundaries goes back to Carlile's threefold typification. According to this, an effective boundary object establishes at a *syntactic boundary* "a shared syntax or language for individuals to represent their knowledge" [16]. At a *semantic boundary*, an effective boundary object "provides a concrete means for individuals to specify and learn about their differences and dependencies across a given boundary" [16]. At a *pragmatic boundary*, an effective boundary object "facilitates a process where individuals can jointly transform their knowledge" [16]. As can be seen in the table, quantitative reporting

	Quantitative	Qualitative	
	Syntactic	Semantic	Pragmatic
Product and portfolio responsibility	Business metrics, stakeholder happiness	User stories, epic status, milestones	Portfolio wall, strategic benefits
Development responsibility	Velocity, test results, sprint burn-down	Highlights	Working software, dependencies map
Process responsibility	Team happiness, impediments in percent	Team radar	Retrospectives, work agreement

Table 2.3.: Qualitative and quantitative reporting with examples of boundary objects [108]

occurs mainly at the syntactic boundary. Qualitative reporting, in contrast, is associated with semantic and pragmatic boundaries [108].

2.3.3. Metrics in agile software development

The seventh principle of the Agile Manifesto states: *"Working software is the primary measure of progress."* This principle indicates the importance of measurement in agile software development, similar to classic project reporting or traditional software development. *Measurement* is defined as:

> "the process by which numbers or symbols are assigned to attributes of entities in the real world in such a way so as to describe them according to clearly defined rules" [35].

Unlike traditional measurement programs, however, the agile approach emphasizes progress measurement through working products rather than measuring the intermediate status of work-in-progress. These different approaches collide in two respects. First, the traditional approach measures progress against predefined plans and measurable goals, while agile metrics embrace change. Second, measurement in the traditional sense is done with an extensive set of metrics, which contradicts the agile principles of simplicity and speed [62]. Kupiainen et al. summarize the differences between traditional and agile measurement this way: The traditional approach includes controlling, outsider viewpoint, tracking deliverables, setting measurable goals, following a plan, large programs. The use of metrics in the agile context concentrates on team control, customer focus, simplicity, following a trend, fast feedback, responding to change. Although the orientation differs, the authors conclude that metrics are used for similar purposes in both worlds [62]. According to Fenton and Bieman, metrics satisfy the information needs by understanding, controlling, and improving software development projects [35]. Among the areas of use are sprint and project planning, where metrics are applied to prioritize tasks and scope activities. Metrics also help monitor sprint and project progress, identify problems, and predict the end result by making the current status transparent. Central to agile principles are continuous improvement and learning, which metrics promote along with motivating people. Some metrics also aim to identify or predict problems in order to solve or avoid them. The final area of use for metrics is the quality measurement before and after software releases [62].

Table 2.4 depicts the five most important and influential agile software metrics based on Kupiainen et al.'s systematic literature review. One of the most important metrics is *velocity*, which, depending on the definition, measures how many feature points, story points, scenarios, or issues are developed per period. The *effort estimate* indicates how many working hours are planned per story. The *customer satisfaction metric* usually indicates how satisfied the customer is with the delivered product on a scale from not at all satisfied to very satisfied. The *defect count* measures the number of defects that occurred for a specific product over a particular period. *Technical debt* measures the effort it would take to fix all the problems that increase technical debt. All of these metrics have three things in common. First, they are easy to use and build on existing tools. Second, they are capable of sparking discussion. Third, they can make problems transparent. In the table, the metrics are assigned to different categories, which several authors have proposed. Fenton and Bieman's categorization includes two dimensions: *entities* and *attributes*. Possible types of entities include *processes* as a collection of software-related artifacts, *products* as artifacts or deliverables, *documents* resulting from a processing activity, or *resources* as entities needed for a processing activity [35]. This classification is reflected in column two. For the five selected metrics, only the types of products and processes apply. Nevertheless, even for all the other metrics examined by Kupiainen et al., it is apparent that measurement focuses more on products and progress and less on human resources [62]. Furthermore, Fenton and Bieman's categorization reflects that the processes of implementation, testing, and the entire development cycle are mostly measured. In contrast, requirements engineering, specification, and design are rarely represented in the measurement of agile software development [62]. In column three, the metrics are classified according to Krebs's cornerstones of agile project reporting mentioned in subsection 2.3.1. The metrics are assigned to two of the three cornerstones, namely *progress* and *quality*. The selected metrics do not cover *team morale*. Subsection 2.3.1 also mentions the five metrics types as a result of Stettina and Schoemaker's cross-case analysis. The selected metrics can be mapped to *performance* and *progress metrics* as well as *quality* and *status metrics*. The areas of use of the metrics are very diverse based on Kupiainen et al. categorization, as the last column shows.

The analysis of 12000 software projects from different countries shows that especially high-performing software companies rely extensively on software productivity and quality metrics, while low-performing ones tend to do not [51]. Productivity is often measured by velocity, while quality is assessed by metrics such as defect count or customer satisfaction. According to Kupiainen et al.'s study, companies use metrics suggested by agile literature and complement them with custom metrics. They also use metrics situationally when a specific problem needs to be solved. The metrics generally follow the principles of the Agile Manifesto. For example, metrics strengthen collaboration and direct communication among agile teams (see principles four and six of the Agile Manifesto in 2.1), which is reflected by

Metric	Type of entity by [35]	Cornerstones categories by [60]	Categorization by [107]	Main area of use by [62]
Velocity	Processes	Progress	Performance and progress	Sprint and project progress tracking and motivating people
Effort estimate	Products	Progress	Performance and progress	Sprint and project planning and progress tracking
Customer satisfaction	Products	Quality	Quality and status	Post-release quality assessment and motivating people
Defect count	Processes	Quality	Quality and status	Pre-release quality assessment
Technical debt	Products	Progress and quality	Progress, quality, and status	Fixing software process problems

Table 2.4.: Categorization of the most important agile software development metrics [62]

the fact that metrics serve as a common basis for discussion and increase visibility. Likewise, metrics motivate employees to act and, thus, contribute to a sustainable working environment based on trust, as stated in principles five and eight. Metrics also make it possible to uncover problems and support continuous improvement and learning (principle twelve). Using specific metrics such as customer satisfaction, defect count, or velocity allows measuring progress and customer satisfaction with working software at regular intervals (principles one, three, and seven). Metrics also supports the ability to react to changing customer requirements in Sprint Planning (principle two). The measurement with minimal effort of technical debt can improve design quality and promote excellent architectures (principles nine, ten, and eleven) [62]. Davis breaks down the principles of the agile manifesto into four focal points. Accordingly, the following questions should be asked when measuring agile teams: Are the *requirements* effective? Is the *development team* effective? Are your *processes* effective? Is your *software* effective? The requirements can be measured, for example, by the metrics recidivism rate, lead time, development time, and velocity. The metrics lead time, development time, deployment frequency, and good/failed builds can assess the development team. Process metrics include velocity and the number of successful deployments. The effectiveness of software is reflected by successful versus failed builds, business metrics, and performance health data [24].

2.3.4. Metrics in large-scale agile software development

Large-scale agile software development teams often use metrics in conjunction with frameworks. Frameworks such as DAD, Scrum-at-Scale, and SAFe provide concrete guidance for the use of metrics. Other frameworks, however, do not explicitly mention large-scale agile

Goal	Metric
Employee engagement	Employee survey; HR statistics
Customer satisfaction	Net promoter score survey
Productivity	Feature cycle time
Agility	Release predictability measure
Time-to-market	Number or releases per year
Quality	Defect data and support call volume
Partner health	Partner and vendor surveys

Table 2.5.: Lean portfolio metrics [66]

metrics [32]. DAD suggests a mix of business-related and agile-related metrics to measure improvement. Among these metrics are, for example, iteration burn-down, business value delivered, Net Present Value (NPS), defect trend, and stakeholder satisfaction [5]. Scrum-at-scale does not mandate a specific set of metrics; nevertheless, it suggests measuring a minimum of metrics. Organizations using Scrum-at-Scale are encouraged to measure the *productivity* of their teams via, for example, the amount of change in the working product delivered per Sprint. Likewise, the framework proposes to track the *value delivery* via, for example, business value per unit of the team effort. In the third area, Scrum-at-scale recommends capturing *quality*. This can be done, for example, via metrics such as defect rate or service downtime. The bare minimum of metrics also includes *sustainability*, measured via team happiness, for example [110].

SAFe provides the most comprehensive set of metrics of any large-scale agile framework. It counts 25 metric types for all four metric levels of the framework: portfolio, large solution, program, and team. SAFe offers Lean Portfolio Metrics, Portfolio Kanban Board, Epic Burn-up Charts, Epic Progress Measures, Enterprise Balanced Scorecard, and Lean Portfolio Management Self-Assessment at the portfolio level. The Lean Portfolio Metrics aim to measure the performance and progress of an entire portfolio effectively. Table 2.5 shows the metrics presented in Leffingwell's SAFe Reference Guide 4.5 [66]. The Portfolio Kanban Board enables the weighting and analysis of epics before the PI boundary is reached. In addition, this metric facilitates prioritization, progress tracking, and quality assured control of epics. Progress tracking is also a particular focus of the Epic Burn-Up Chart. Here, progress can be visualized at the portfolio level using three lines: initially estimated story points, actual story points completed, and cumulative story points completed. The Epic Progress Measure is a visual report that provides an overview of the status of all epics in a portfolio. The report shows the epic name, the sum of the currently estimated story points, the initial estimate of the Epic in story points, and the ratio of current estimated to initial estimated story points. The Enterprise Balanced Scorecard divides into Efficiency, Value Delivery, Quality, and Agility. An executive dashboard then displays the corresponding metrics. Lean Portfolio Management bases on a self-assessment questionnaire that periodically assesses performance, intending to improve the methods used by an organization. An auto-generated radar chart illustrates the relative strengths and weaknesses of portfolio practices [66].

At the large solution level, there are three metric types in SAFe: Solution Kanban Board, Solution Train Predictability Measure, and Solution Train Performance Metrics. The Solution Kanban Board facilitates the flow of capabilities through the continuous delivery pipeline by visualizing the current status in the life cycle. The Solution Train Predictability Measure summarizes the ART's predictability measures and displays them graphically. The y-axis shows the degree of achievement of the objectives in percent. The x-axis displays the individual PIs. In addition, a tolerance range is drawn. The Solution Train Performance Metrics also summarizes the performance metrics of the ARTs. The metrics contained encompass the areas of functionality and quality. Functionality metrics include program velocity, predictability measures, as well as number of features, enabler features, and stories, respectively planned and accepted. The quality metrics comprise the unit test coverage in percent, the number of defects and total tests, as well as the number of automated tests in percent and non-functional requirement tests [66].

In SAFe, the following twelve metric types can be used at the program level: Feature Progress Report, Program Kanban Board, Program Predictability Measure, Program Performance Metrics, PI Burn-Down Chart, Cumulative Flow Diagram, Agile Release Train Self-Assessment, Continuous Delivery Pipeline Efficiency, Deployments and Releases per Timebox, Recovery over Time, Innovation Accounting and Leading Indicators, and Hypotheses Tested over Time. The Feature Progress Report tracks the status of features and enablers during PI execution. It highlights which features are on track or behind plan at any given point in time. Unlike the Solution Kanban Board that focuses on capabilities, the Program Kanban Board visualizes the current process state of features. The Program Predictability Measure, analogous to the Solution Train Predictability Measure, consolidates the team PI performance reports. The Program Performance Metrics measure the performance of an ART at the end of a PI and contain the same metrics as the Solution Train Performance Metrics. The PI Burn-Down Chart shows the progress measured in story points of each iteration in the PI time window without revealing the progress of each feature (the task of the Feature Progress Report). It plots the planned work for a PI against the actual accepted work "burned down" on two lines. The Cumulative Flow Diagram partitions the number of features into different ranges of Kanban states using lines. Typical Kanban states include Funnel, Analyzing, Backlog, Implementing, Validating on staging, Deploying to production, Releasing, and Done. This metric visualizes the overall progress of features and the identification of potential bottlenecks in the states. Agile Release Train Self-Assessment allows for reflection on the progress and performance of the ART via a self-assessment questionnaire that the ART Engineer fills out. The visualization is done via a radar chart with different dimensions and a 0 to 100 scale. Continuous Delivery Pipeline Efficiency compares touch time to wait time relative to each stage of the pipeline cycle. Touch time measures the time it takes teams to add value, which is typically a small fraction of total production time. The Deployments and Releases per Timebox metric type aims to show and increase the frequency of deployments and releases. It can be viewed on a PI basis or more frequently, e.g., even in the middle of a PI. Recovery over Time is a report that measures the number of rollbacks that occurred either physically or by turning off function toggles. Also displayed is the date of release or deployment to identify dependencies between the two. The Innovation Accounting and Leading Indicators help quickly run experiments and give the customer the ability to validate hypotheses. Both Minimal Marketable Feature (MMF)s and Minimal Viable Product (MVP)s serve the definition of leading indicators so that progress toward the benefit hypothesis can be measured. The last of the metric types on program levels is Hypotheses Tested over Time. This metric is entirely in the spirit of hypothesis-driven development and visualizes the number of hypotheses verified versus failed in a PI [66].

At the lowest level, SAFe proposes the metrics types: Iteration Metrics, Team Kanban Board, Team PI Performance Report, and SAFe Team Self-Assessment. The Iteration Metrics cover the same metrics measured in the Solution Train and at the program level, with the difference that they only refer to a single team. The Team Kanban Board can be kept leaner than the Program Kanban Board (e.g., define, analyze, check, create, integrate, test). In the Team PI Performance Report, business owners, customers, agile teams, and other key stakeholders rate the actual business value achieved for each team's PI objectives. Ideally, trains should be in the 80 to 100 percent range to enable effective planning. SAFe offers a self-assessment also at a team level, where teams continuously assess themselves, visualize their strengths and weaknesses via the radar diagram, and, thus, try to improve their processes [66].

2.3.5. Goal-based measurement

Three approaches have emerged in the prior literature to understand value creation, its measurement, and the derivation of goals in the agile context: strategic business objectives [52, 56], stakeholder needs [69, 70], and agile principles [24, 62]. The first of the three approaches aligns measurement with the business interests of customers and investors. With the procedures of the Balanced Scorecard by [52] and the Software Value Map by [56] that builds on it, strategic business objectives measure themselves in the dimensions: *customers*, *financial*, *internal business*, and *innovation and learning*. Although the Software Value Map is designed for software development, there is no particular focus on agile methods [58]. An extended approach to goal-based measurement is the stakeholder-driven measurement, which includes customers, investors and employees, and society [58, 83]. The derivation of goals for measurement in the agile context can also be carried out based on the Agile Manifesto principles, as described in subsection 2.3.3. Korpivaara et al. summarize the performance dimensions that stem from the three approaches: *financial value*, *customer value*, *internal process efficiency*, *collaboration and culture*, and *innovation and learning* [58]. In Table 2.6, these five dimensions are presented together with examples of goals for agile software development organizations (adopted by [58]). Some of the example goals match the Lean Portfolio Metrics goals of Table 2.5.

The literature provides several frameworks, models, approaches, and standards to help software organizations to identify goals from various stakeholder perspectives and implement the associated measurement programs. These include Balanced Score Card [52], GQM [8], Model Measure Manage Paradigm [82], Goals Questions Indicators Measures [14], Decision Maker Model [72], Measurement Information Model in ISO/IEC 15939 Standard [2002], and GQM+Strategies [7, 94]. The core idea behind most of these approaches is to connect

Dimension	Examples of goals
Financial value	Return on investment, development cost
Customer value	Customer satisfaction, product quality, product usability, delivery time, delivery quality
Internal process efficiency	Productivity, predictability, development quality, responsiveness to change
Collaboration and culture	Employee satisfaction, collaboration, acceptance of requirement changes, customer care, agile tool usage, self-organization
Innovation and learning	Capability development, continuous improvement

Table 2.6.: Performance dimensions and their goals for measurement in the agile context [58]

metrics to organizational goals, strategies, and/or team information needs, thus establishing cost-effective measurement programs by collecting only valuable data with clear intentions. This idea follows the principle of simplicity of the Agile Manifesto as well. Nevertheless, most of these practices were introduced when agile software development was not as widespread as today. In recent years, however, there has been an increase in research regarding the use of the GQM method and its variations in the context of goal-driven metrics selection in agile software development [13, 90, 91]. The approach, proposed by Basili and Weiss back in the 1980s, has become a quasi-standard. It makes it possible to define goals and systematically refine them into measurable values without disregarding the context and following a predefined plan. In contrast to a pure bottom-up approach, the GQM method always starts top-down with the goal refinement process and concludes with a bottom-up data analysis and interpretation phase [94].

Overall, the method divides into four phases: planning, definition, data collection, and interpretation (see Figure 2.5). At the beginning of the *planning* phase, the GQM team is assembled, the desired area of improvement is specified, and the object of investigation is selected and characterized. The result is a project plan that sets out the organizational structure, a schedule of measuring, and the necessary training for the people involved in the measurement process. The subsequent *definition* phase initially identifies the measurement *goals* on a conceptual level through interviews or derives them from the corporate strategy. The goals have to be documented in a structured way. The use of the following template can help. Goals define a purpose (e.g., achievement, improvement, control), for the object under investigation (e.g., test process, software product), for a specific issue or quality attribute (e.g., reliability, quality, speed), from a perspective (e.g., user, developer), and within a certain context (e.g., people involved, environment, resources, organizations) [59]. For example, a goal might be:

> "Improve (=purpose) the quality (=issue) of software product X (=object) from the viewpoint of customer Y (=perspective) within development department Z (=context)."

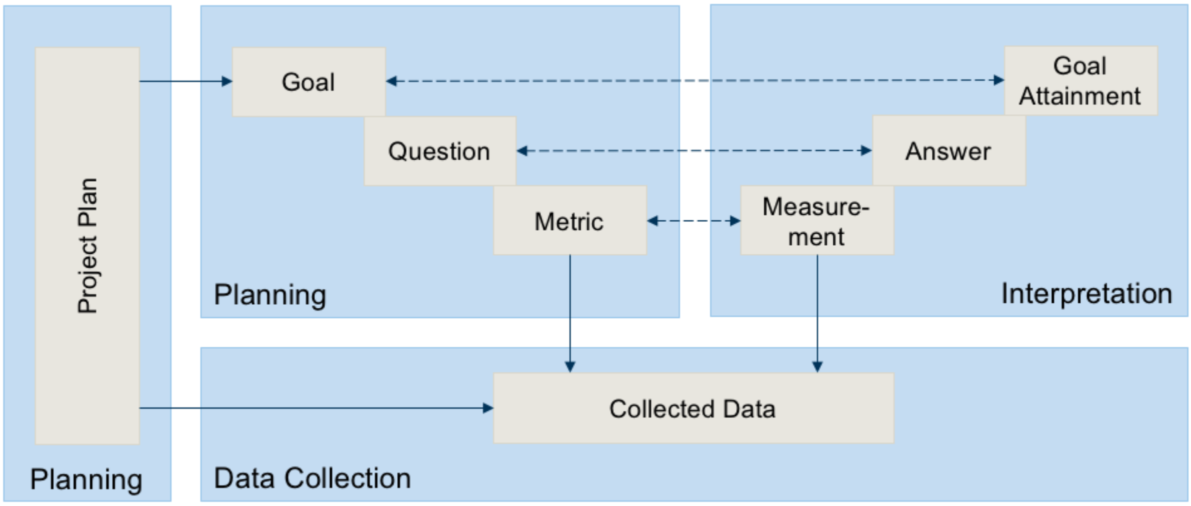

Figure 2.5.: The four phases of the GQM-method [103]

Building on the formulation of the goal, *questions* are developed that refine the goal and, thus, make them operational. The questions can be specified either qualitatively or quantitatively and are answered accordingly using either a questionnaire or metrics. It is also advisable to formulate a hypothesis for each question to increase the learning effect when comparing knowledge before and after measurements. After defining goals at the top level and questions at the intermediate level, the definition of the corresponding metrics follows at the lowest level. At least one metric should be defined for each question and continuously checked for consistency and completeness. The results of this definition phase lead to a GQM plan and also include a measurement plan and an analysis plan. The actual measurement takes place in the *data collection* phase. The final *interpretation* phase processes the collected data according to the previously defined metrics to obtain measurement results. The GQM team can use the measurements to answer the questions. With the answers, in turn, it can evaluate whether the original objectives have been attained [59].

Over time, the GQM+Strategies approach expanded the classic GQM method by incorporating organizational goals, strategies, and data into measurement programs. This evolution allows for the comprehensive alignment of organizational goals and strategies across different units of an organization. In addition, GQM+Strategies focuses on providing concrete rationales for why goals and strategies are linked across different units. This procedure results in a much more comprehensive and complex measurement program that consists of several measurement levels and additional process phases compared to the classic variant. The benefits lie in improved alignment, communication, and decision-making between the different business units [94].

2.3.6. Metrics team maturity model MeTeaM

MeTeaM is a model by Meding et al. for assessing the maturity of software metrics teams and their role in designing, developing, and maintaining measurement programs in an organization. The model supports software metrics teams in quantifying their maturity level across multiple dimensions by using a template and, thus, uncovering the areas of improvement. According to Meding et al., software metrics teams can be dedicated teams or virtual teams whose members distribute among the software development teams. In the agile context, the members of software metrics teams are usually RTEs/Scrum Masters, Product Owners, Product Managers, or developers, responsible for planning, defining, developing, maintaining, and operating measurement programs [73]. Roles on the software metrics team include, for example, measurement analyst, measurement designer, and measurement librarian. Measurement programs comprise two levels: the technical measurement system and the social actors and their organizations. Therefore, measurement programs are also referred to as socio-technical systems. They build on a measurement infrastructure that incorporates the metrics team, databases, information products, and measurement systems. Measurement programs provide information to sponsors, stakeholders, and users, who leverage it for decisions, insights, and early warnings [105].

The researchers designed and evaluated the MeTeaM model using the action design research method with four metrics teams from four distinct case organizations in the field. All case organizations work according to agile principles in a scaled setting, partly within frameworks like SAFe. Two employ dedicated teams, and two employ virtual teams. The case organizations cover both small and large companies. Observations of the metrics teams' needs and insights from a literature review synthesize the model in six areas, each subdivided into organizational and team maturity levels [73]. The model's template includes several statements for each of the six areas listed in Appendix A.2.

The first area - *set-up* - addresses the formal structure, mandate, and roles of the metrics team as well as its organizational context. Ideally, the metrics team should have a formal assignment from the organization and be recognized by the organization [73]. The elements that shape the area set-up can be found with their associated statements in Table A.1. The first sub-area, formalization, is guided by Staron and Meding findings regarding a team's assignment, resources, ways-of-working, vision, and accessibility [105]. The second and third sub-areas focus on formalized measurement-related roles in the software metrics team and the organization according to the ISO/IEC 15939 standard. The second area of the model is about the *commitment* of the metrics team on the one hand and the management of an organization, on the other hand, to integrate measurement data into the decision-making processes. The area refers to the degree of commitment towards standards and processes. It also provides visibility into the alignment of team goals and organizational goals [73]. The application of Staron and Meding's [105], Sandberg et al.'s [97], and Gopal et al.'s [38] findings, reflected in the statements of Table A.2, are critical to the success of the measurement program and the creation of value. The third area describes the *impact* of the measurement program on the organization, its products, processes, and customers. The metrics team aims to impact these stakeholders via measurement results in the form of information products such as dashboards, charts, or excel files to recognize its value. The model distinguishes between the sub-areas impact on the organization and impact on outside/outreach [73]. The underlying findings of the statements in Table A.3 originate mainly from Staron and Meding [105] and Kilpi [57]. The fourth area - *longevity* - is about the ability of the metrics team to evolve and adapt to organizational change over time. Longevity means continuous value contribution and a proactive attitude to reinvent itself again and again [73]. The key findings of the longevity statements in Table A.4 stem from Ferris [36] and Krell [61]. The fifth area expresses itself in the metrics team's ability to quickly and reliably transform information needs into high-quality information products, demonstrating an *efficient way of working*. Moreover, the organization should be able to use new metrics in its decision-making processes and compare different metrics in meaningful ways [73]. Table A.5 provides an overview of the statements from Staron and Meding [105]. The sixth area revolves around the ability of the team members to work together respectfully, deliver a high-level *team performance* while keeping the well-being of the team in mind [73]. The area includes statements in the sub-areas of context, mission, talent, norms, buy-in, resources, and courage, as shown in Table A.6. The insights incorporated into the statements derive from Curphy's [23], Espinosa et al.'s [33], and Liang et al.'s [67] findings, among others.

2.3.7. Measurement challenges

When measuring the achievement of goals through metrics in the context of large-scale agile software development, some challenges can arise. Korpivaara et al. report that one of the most frequent challenges results from varying goals at different organizational levels. Agile teams may feel disconnected with their work effort from portfolio-level goals if the measurement structure provides few metrics that contribute to the portfolio metrics. This challenge is compounded when teams feel that powerful external factors outside their control reduce their contribution to portfolio outcomes [58]. Oza and Korkala come to a similar conclusion. According to them, visibility over the teams' influence is very important to gain trust and control [83]. Furthermore, limited availability of data, time, expertise, and guidance prevents agile teams from benefiting from metrics because it forces them to limit the choice of metrics [58, 90]. Another challenge is standardizing measurement practices to make decisions based on robust data and to make teams comparable. Best practices to deal with the challenges propose reviewing and re-prioritizing the metrics constantly and talking about them regularly to create a sense of motivation and purpose within teams [58].

3. Related work

This chapter presents and summarizes the key publications related to the topics of this thesis. First of all, this includes the studies that deal with the adoption of metrics and strategic goals in agile software development. The second section highlights a publication central to this thesis, which is one of the few that examines the adoption of metrics and strategic goals in a large-scale context. Overall, the related work extends and deepens the foundations presented in the last chapter.

3.1. Adoption of metrics and strategic goals in agile software development

Boerman et al. (2015)

Boerman et al. devise a metrics-based reporting model for monitoring agile development processes to create an improved sense of control and facilitate communication between developers, managers, and sponsors. For this purpose, they adopt the GQM approach and derive a set of eleven metrics from the project owner's goals. The metrics mainly access data from the product backlog. A case study verified the practical feasibility, reliability, and usefulness of the model. Respondents in the study reported being able to link actual project events to metric numbers. The value of the usefulness of each metric varied. For example, the estimation shift and enrichment rate metrics were particularly valuable to the customer, while the changed product backlog items metric received low agreement [13]. Similar to this thesis, researchers employ a goal-oriented approach to identify agile metrics.

Ram et al. (2018) & Ram et al. (2019)

Ram et al. conducted several case studies as part of the Q-Rapids project to investigate which metrics software-intensive companies use with which rationales to assess agile software development processes [90]. Furthermore, they observed both challenges [90] and success factors [91] in the operationalization of agile process metrics. To collect the data, the researchers held several workshops applying the GQM approach. The study reveals that the case companies are primarily interested in velocity, testing performance, estimation accuracy, code quality, development speed, delivery performance, and external quality. The metric selection is justified by data availability, tracking of planning and bottlenecks, traceability, control, awareness creation, and knowledge sharing [90]. According to Ram et al., the unavailability of data and the lack of ability to extract actionable information from metrics can hinder the operationalization of metrics [90]. Hence, data availability and alignment with the development process are fundamental to the success of metrics implementation programs in an agile context. The authors also report that the operationalization should be aligned with the iterative way of working to enable long-term success. Furthermore, regular feedback and a trust-building approach to metrics are essential [91]. With the challenges and success factors identified, this research in the agile context forms a valuable basis for introducing metrics at a scaled level in the course of this thesis.

Cheng et al. (2009)

Another relevant aspect of this thesis relates to agile software development governance, which involves monitoring and control mechanisms such as metrics. The paper by Cheng et al. presents a list of KPIs and interventions that software development managers can use to control agile development processes. Both lists stem from three case studies. The first list includes classic KPIs such as team effort remaining, weekly working hours of individuals, number of completed tasks, and test success rate. These measurements provide information about the achievement of goals and indicate deviations. If deviations occur, there are possible means of intervention at different levels. For example, at a team level, an additional team or additional work can be assigned to a team. Intervention can be carried out at an individual level by assigning a person with more experience and expertise to assist. In addition, tasks can be moved to subsequent sprints or escalated to cross-team decision-making levels. In terms of quality, criteria for working software can be defined [17].

Oza and Korkala (2012)

The focus of this research is to investigate the adoption of agile metrics based on a case study. Oza and Korkala provide six concrete recommendations to improve measurements in agile software development, which supports the adoption of metrics in the context of this thesis. Accordingly, it is vital to make the impact of the metrics visible at all organizational levels. They also recommend intensifying the collective vision on agile metrics to create a culture of measurement. Additionally, the researchers emphasize understanding the collected metrics' reason, relevance, and importance and limit the number if necessary. An imperative tool in the presentation of metrics is the dashboard, which visibly highlights the evolution of indicators for all employees. It is also clear from the case study that teams require a holistic view of what is happening to avoid slowdowns or technical debt caused by unforeseen tasks. For successfully building a measurement culture, it can also be helpful to pursue just one metric for a given period rather than following too many metrics. Oza and Korkala note that the perceptions by various stakeholders on the impact of agile adoption warrant further solid evidence. The researchers conclude with an appeal for designing a frame of reference of metrics programs [83].

Choras et al. (2020) & Coyle and Barata (2016)

Even though the content of both articles differs slightly, they both follow the principles of the action design method similarly as this thesis aims to do. The main contribution of Choras et al. lies in the validation of the adoption of a process metric catalog in the context of agile software development at a medium-sized IT provider. In the first phase, the action design team defined 25 metric candidates based on principles and observations shared by the Scrum team. The metric candidates include general metrics such as the number of development tasks, task estimation metrics such as effort estimation accuracy, task implementation metrics such as average time-to-implementation, bug fixing metrics such as the number of development tasks with reported bugs, and testing metrics such as percentage of tasks waiting for testing. Next, the action design team assessed the candidate metrics regarding process performance, product quality, and task estimation. In consultation with the Scrum team, the action design team selected nine metrics that proved more valuable than the others. Overall, the implementation of the metrics improves reporting of actual effort and the comparison to the planned effort. This improvement enables greater estimation accuracy. In addition, the efficiency of tracking tasks has increased, and the Scrum team could identify bottlenecks more quickly [18].

Coyle and Barata's action research project proposes a model for socially constructed metrics that helps organizations create agile quality. The model consists of three different types of socially constructed metrics: people-related, process-related, and outcome-related. Moreover, the model assesses introduced metrics in terms of three indicators: evidence, a comparison with past results; expectations, a comparison with the expected outcome according to the original plan of the stakeholders; and evaluation, a critical analysis by the agile team. Each metric is evaluated against the three indicators from 0 to 100 and results in a weighted average. Based on the result, agile teams can decide whether quality improves by implementing the particular metric. In addition, the model enables a contextualized perspective on quality, transfer of tacit knowledge; critical reflection on quality; and effective support in daily meetings, retrospectives, and audits. It also proposes a participatory approach to continuous improvement in agile software development [22].

3.2. Adoption of metrics and strategic goals in large-scale agile software development

Korpivaara et al. (2021)

Korpivaara et al.'s exploratory case study addresses the existing knowledge gap regarding performance measurement and its challenges in scaled agile organizations. Within the study, the authors examined two development units of an organization, both formally using SAFe. Moreover, metrics and goals were analyzed separately at the portfolio level and the program and team levels. Customer satisfaction stands out as the most important goal at the portfolio level, measured by the Net Promoter Score and Customer Experience Index metrics. Another important goal at this level is financial value, which comprises income and costs. The former can be measured by generated income, revenue, and sales per service channel. The latter divides again into the goals of cost budget compliance and activation of cost savings. The study's researchers report IT stability as a third portfolio-level goal. The metrics number of reported incidents and application downtime were used for measurement here. The program-level goals are directly derived from the portfolio-level goals. Hence, the measurement purpose shifts from the outcome (what) to the efficiency of internal processes across teams (how). The goals at this level follow the recommendations of the SAFe framework, as shown in Table 2.5. Thus, the case company possesses the goals: productivity, time-to-market, quality, continuous improvement, customer satisfaction, employee engagement, and alignment with strategic goals. Productivity aims to reduce feature cycle time and can be measured primarily by the number of completed features and iteration velocity. Time to market is represented by the metric feature lead time, which spans from ideation to delivery. The case organization uses the number of defects and the test automation rate to measure quality in development. Quality in production is measured by the number of business-critical incidents and the mean time to recovery. The other goals, such as continuous improvement, employee engagement, and customer satisfaction, are measured and complemented by qualitative self-assessments and satisfaction surveys. The seventh and final goal intends to measure the progress and impact of the features worked on by the teams. Therefore the Objectives and Key Results (OKR) framework was introduced to define the metrics [58].

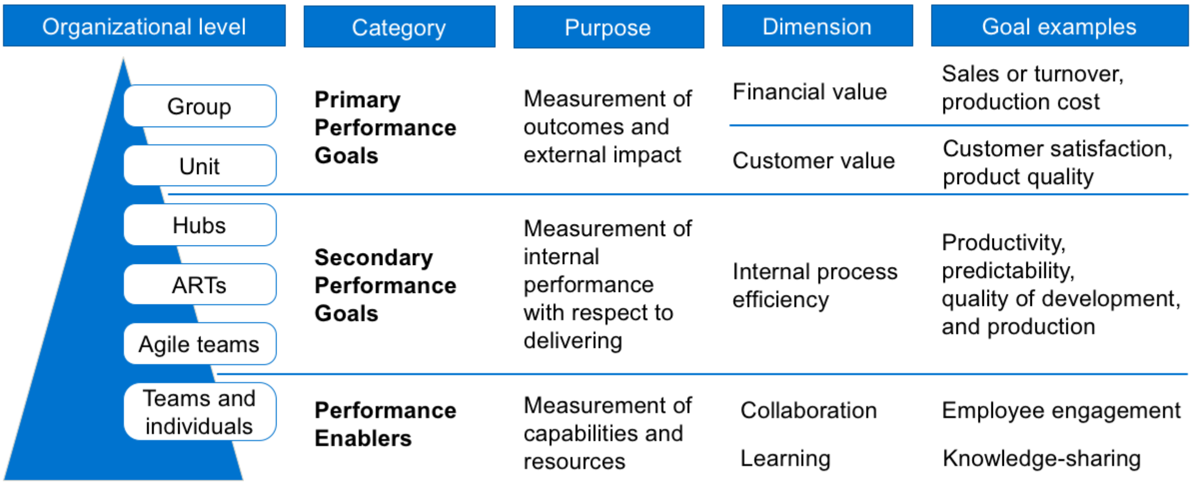

Figure 3.1.: Structure of performance goals in scaled agile organizations [58]

Korpivaara et al. summarize the collected findings in Figure 3.1. It shows that financial value and customer value are the main focus at the group and unit level and are measured by portfolio metrics. At the secondary level, the measurement of internal process efficiency plays a key role in achieving the primary performance objectives. The third level has only an indirect influence on the primary goals. Nevertheless, teams measure the performance dimensions of collaboration and learning because the impact on the secondary goals is enabled, and alignment between teams is increased [58]. Overall, Korpivaara et al.'s study has the most overlap with this thesis because it is the only one that has explicitly examined agile metrics at the scaled software development level.

4. Case study

This chapter describes the case study that was conducted as part of this thesis. The first section is devoted to the case study design, which contains all the essential elements of a case study plan. The second section describes the case and its two units of analysis. The following five sections present the case study results and, thus, answer the five corresponding case-related sub-questions. Section 4.3 describes the identified goals in the context of existing metrics. Section 4.4 addresses the existing metrics. Section 4.5 then reports on the distribution of responsibilities for metrics and goals. Section 4.6 covers the challenges of introducing and using metrics. Finally, Section 4.7 summarizes the recommendations for metrics adoption from the interviews.

4.1. Case study design

The case study builds on the proposed guidelines for case study design by Runeson and Höst [95], as described earlier in Section 1.3. Accordingly, a case study plan was formulated, which determines the structure of this section in its elements. This detailed description documents the methodological approach of the case study and informs the reader about the most relevant design decisions.

Objective — what to achieve?

The primary objective of this case study is to accompany and explore the goal-oriented adoption and management of metrics in the large-scale agile software development context. For this purpose, the first step must capture the current situation and derive the need based on this foundation. The focus is initially on the metrics used in the past and their associated goals. In the research, the explorative character increasingly becomes an ameliorative one, as the goal-driven introduction of additional metrics improves the current state. The insights gained are intended to be used in conjunction with the findings from existing research to create a solution artifact within the scope of this work, thereby ensuring the practical relevance and value of this resulting solution artifact.

The case — what is studied?

The case organization under investigation is an insurance group that provides software solutions to various business areas through specific organizational units. Two of the case organization units are part of the case study and are considered separate units of analysis (see Figure 4.1). According to Yin, this makes the case study an *embedded case study* [117].

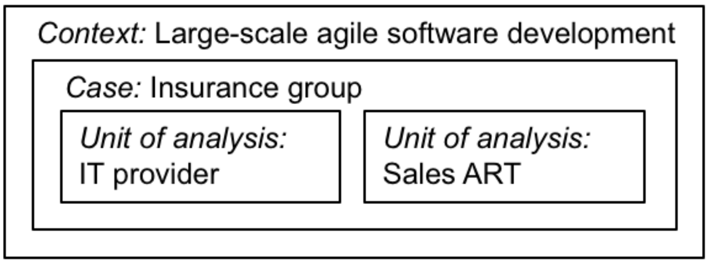

Figure 4.1.: Embedded case study setup

The first unit of analysis is a German IT-provider for the insurance industry. The second one comprises the Agile Release Train (ART) for the sales department of the case organization. In both cases, the units of analysis include the entire agile development organization and its associated management structures, which consist of teams and individuals with their respective roles. The thesis analyzes the present case in particular from the perspective of scaled software development. The case organization is a suitable and interesting research object because the units of analysis differ in their scaled framework and maturity level. The IT-provider has comparatively young teams that intend to introduce additional metrics on a cross-team level. The Sales ART, in contrast, has more traditional and larger structures where metrics are deployed at a scaled level. Further details on the case organization are described in more detail in Section 4.2.

Theory — frame of reference

The theoretical frame of reference for the research is formed by the previous scientific contributions to metrics in the two areas of agile as well as large-scale agile software development. The foundations in Chapter 2 and the related literature in Chapter 3 make the context of the case study research clear and help to verify the results later on.

Research questions — what to know?

The research questions indicate what needs to be known to fulfill the study's objective [95]. This objective of the study stems from the research questions of the thesis presented in Section 1.2. The case study mainly aims at answering the first of these research questions and specifies it in the four following case-related sub-questions:

1. With what goals are metrics being used in the case organization so far?
2. Which metrics are used and how?
3. How are responsibilities for metrics and goals distributed?
4. What are the challenges associated with the introduction and usage of metrics?

The case study only briefly addresses the second research question of the thesis through the following fifth sub-question:

5. What recommendations do stakeholders have for metrics adoption?

All five case-related questions are reflected by the semi-structured interview questionnaire, accessible in Appendix A.1. As explained in Section 1.3, the case organization environment also serves within the action design research the continuous development and evaluation of the goal-based metrics introduction and management approach, described in more detail in Chapter 5. The execution of the approach follows in Chapter 6, where the first two steps of the metric adoption approach partially overlap with the case study sections 4.3 and 4.4. The formal evaluation of the approach also takes place in the context of the case study organization. This evaluation is described in more detail in Chapter 7.

Methods — how to collect data?

Using multiple data sources and perspectives of different roles within the case study is vital to achieving a differentiated and more meaningful result [95]. Therefore, the following methods were applied:

- Informal interviews
- Semi-structured interviews
- Agile meeting observations
- Document analysis
- Surveys

The informal and unstructured interviews usually took place weekly during the practical part of the work. In particular, I spoke extensively with the two contacts from the two analysis units (SM1, RTE1) of the case organization. The informal interviews during the initial period mostly revolved around the investigation of the problem domain and the metrics introduction planning. As the research progressed, the emphasis increasingly shifted to the organizational setting and the proper design of the approach. This concerned, for example, details of existing and new metrics, prioritization of goals, or integration into agile frameworks. In the meantime, the focus was permanently on the feedback for the artifact and the refinement of the requirements. Both were of enormous importance for the artifact evolution in terms of action design. In addition, I conducted formal, semi-structured interviews with IT employees to systematically capture and analyze the current state of the usage of metrics in large-scale agile software development. In contrast to unstructured interviews, in which the researcher formulates the interview questions spontaneously as general concerns and interests, semi-structured questions are planned in advance [95]. In developing the questionnaires, I drew on existing questions from other researchers that have also been used in interviews with other

No.	Main role	Alias	Agile experience	Case unit
1	Release Train Engineer	RTE1	2 years	Sales ART
2	Scrum Master	SM1	10 years	IT provider
3	Product Manager	PM	2 years	Sales ART
4	Developer	Dev1	5 years	Sales ART
5	Product Owner	PO	4 years	IT provider
6	Scrum Master	SM2	6 years	Sales ART
7	Scrum Master	SM3	3 years	Sales ART
8	Principal Product Owner	PPO1	3 years	IT provider
9	Chief Technology Officer	CTO1	11 years	IT provider
10	Information Security Officer	ISO1	4 years	IT provider

Table 4.1.: Interview partners of semi-structured case study interviews

industry partners. The remaining questions arose from the informational interest regarding goals and questions and challenges and recommendations related to metrics adoption. The questions of the semi-structured interviews can be found sorted by the specific topics in Section A.1 of the Appendix. The interviewees cover an extensive range of roles. I conducted ten interviews with employees of eight different roles. This coverage was desired to capture the different perspectives on metrics and goals. Table 4.1 shows the roles of the interview participants, their alias, their experience in agile software development, and their association with the corresponding unit of analysis. The alias helps to assign the statements in the text to the interviews. RTE, SM, PO, and developer belong to the group of people most directly exposed to agile metrics.

Agile metrics exert a less intense but still noticeable influence in their daily work for the other persons. The Product Manager and Chief Technology Officer (CTO) make decisions based on agile metrics. The Principal Product Owner uses agile metrics for effort estimation and contract costing. The Information Security Officer had many touch-points with agile metrics in his previous positions and would like to introduce a new metric. The average experience with agile software development was five years. Experience with large-scale software development varied from one to two years in most cases. Interview time averages nearly one hour, with some interviews taking significantly longer, particularly those with the contact persons. I recorded, anonymized, transcribed, and coded all semi-structured interviews. Transcribing and coding were performed according to Miles et al.'s procedure and recommendations [74]. The tool MAXQDA supported this analysis process. At the beginning of the first coding cycle, I developed and deductively assigned a priori codes to summarize larger segments of the interview text. These 15 a priori codes base on the topics of the interview questionnaire. During coding, I progressively supplemented this preliminary selection with other codes in an inductive manner. In the second coding cycle, I grouped the summarized text segments inductively into smaller categories via pattern coding. This grouping resulted in several hierarchical levels of sub-codes and sub-categories within a differentiated inductive-deductive code system. In the analysis, I evaluated the sub-codes along with the high-level codes, mostly within the analysis unit boundaries. However, I evaluated the codes on the recommendations and challenges in the course of metrics introduction across units of analysis. I used the matrix display approach to evaluate the data items of the metrics code [74].

To describe the existing metrics as accurately as possible, I also resorted to the document analysis method. I also conducted observations at agile meetings, such as the Scrum of Scrum event or demonstrations of tools such as Jira. Furthermore, I applied two surveys to evaluate the approach in the context of the case study. Taken together, the application of the case study methods across all phases of the design science research described in 1.3 expresses the continuous analysis, feedback, and validation of the solution artifact. The final evaluation of the artifact through the surveys describes Chapter 7 in detail.

Selection strategy — where to seek data?

The case organization was intentionally chosen as a follow-up to a series of interviews because we hoped to gain *"revelatory"* insights into the process of selecting, establishing, and managing metrics in a large-scale agile environment. During the interview series, it became apparent that many companies are currently facing the challenge of establishing metrics or reviewing their establishment, making this a *typical* case. Furthermore, the divergent characteristics of the two units of analysis allow for comparisons between the results of actions taken.

4.2. Case description

IT-provider

The first unit of analysis is a medium-sized German IT-provider of software solutions within the insurance group. The IT-provider employs up to 30 people, which work in several distributed teams on several products. This includes about 20 developers and several business analysts, release managers as well as test managers. The analysis unit is organized as a project launched in 2018 to operate and further develop existing software for internal customers. Subsequently, the scope was expanded, and the software was further commercialized to a growing external customer base. This commercialization requires a lot of further development and adaptation to the needs of the customers. Within the project, two pure development teams work together with an infrastructure team. A third development team was formed during the research from existing and new personnel. In addition, another team is in the planning phase. The teams use the Scrum-at-Scale framework (see Subsection 2.2.4) and work in a rhythm of 14-day sprints. The teams live the framework elements more and more and strive to expand scaling as new teams are added.

Sales Agile Release Train

The second unit of analysis organizes itself as an Agile Release Train responsible for developing software for sales. The ART functions across divisions within the case organization and therefore has many interfaces with other development units. The ART comprises 80 employees divided into ten teams, more than 50% of which work in a distributed manner. The teams use SAFe as the agile scaling framework and its Essential SAFe configuration, as described in Subsection 2.2.3. The introduction of agile metrics in a scaled context was started at the beginning of the year. The introduction took place in collaboration between the steering board and agile teams. Metrics were primarily introduced at the program level. At the team level, the use of metrics so far varies greatly and depends strongly on the maturity level of the agile team. For example, some ART teams have previously not used any agile metrics.

4.3. Goals related to existing metrics

This section answers the first research question of the case study by describing to which goals the metrics introduced so far in the case organization relate.

IT-provider

The project as one unit of analysis receives much attention from the board of directors because it helps the company capture an additional business area with significant potential. Therefore, one expectation of the board of directors from the agile development teams is to satisfy their information needs through metrics and, thus, make the project's progress transparent. This expectation links to the goals of achieving the highest possible quality and satisfying the customer in the end. From SM1's point of view, it is also about ensuring ease of use of the software product for the customer. The goals in the customer value dimension encompass transparency as well as commitment in communication. In financial terms, the development organization as a whole aims to save time and, thus, costs by eliminating overhead and coordination efforts.

Concerning individual metrics, the interview partners reported further organization-specific goals. In the case of the metric velocity, the goal is to create an understanding and a feeling among the teams and individuals for planned and achieved work. In addition, from the viewpoint of SM1 and PO1, the goals of planning support and predictability of the required capacities are associated with this metric. The velocity can be used to approximate how many teams or team members are needed for which scope and by when to deliver. A similar goal serves the change velocity, which helps in estimating and planning the effort. The PPO1 can then use this information in communication with customers and management. Once the planned and estimated epics are complete, the metric also helps with billing and cost management. The burn-down chart serves the interrelated goals of making the quality of the requirements definition and the development speed transparent. It intends to reach the ideal line in the burn-down chart. At the same time, the organization aims to attain the goal of increasing the development quantity. For this purpose, the requirements should be well defined and cut as small as possible. Ultimately, this should reduce the time-to-market. Moreover, the scope of a sprint should not increase to any great extent. Therefore, the change in the sprint backlog is monitored, which ultimately promotes transparency over requirement

Org. level	Category	Dimension	Goal
Group	Primary performance goals	Customer value	Customer satisfaction
			Product quality
			Ease of use
			Transparent customer communication
			Commitment in customer communication
		Financial value	Time and cost saving
			Cost management and billing
Scrum of Scrums	Secondary performance goals	Internal process efficiency	Planning support
			Predictability of required capacities
Agile teams	Secondary performance goals	Internal process efficiency	Transparency over requirement quality
			Development speed, time-to-market, and quantity increase
Teams and individuals	Performance enablers	Learning	Understanding of commitment

Table 4.2.: Overview of the IT-provider's metric related goals with classification according to [58]

quality. An overview of the stated goals at the different organizational levels can be found in Table 4.2. The goals classify into Korpivaara et al.'s performance goal structure of Table 3.1 from Section 3.2.

Sales Agile Release Train

In general, the ART pursues general goals with regard to metrics, such as transparency and control over process and effort distribution. The RTE1 mentioned the goals of further development, generating more change volume, and cost reduction for the metric run and change. Cost reduction concerns the financial dimension and, together with the quality goal, is linked to the error tickets metric. Furthermore, in addition to further development and quality of design and development, the organizational unit aims to achieve customer-centricity and reliability in commitments to customers as goals for the metric ART commitment quote. Related to the reliability goal is planning commitment, which ultimately enables reliable commitments to customers. Another goal, which lies behind the metric of the same name, is employee satisfaction. In the future, the organizational unit intends to pursue the goal of quality more strongly. The overview of the ART's goals according to the classification of Korpivaara et al. are shown in Table 4.3.

Organizational level	Category	Dimension	Goal
Group	Primary performance goals	Customer value	Customer satisfaction
			Customer-centricity
			Reliability in the commitment to the customers
Unit	Primary performance goals	Financial value	Cost reduction
Agile release train	Secondary performance goals	Internal process efficiency	Quality of design and development
			Transparency about development processes and effort distribution
			Further development
			Change volume
Teams and individuals	Performance enablers	Collaboration	Employee satisfaction

Table 4.3.: Overview of the ART metric-related goals with classification according to [58]

4.4. Existing metrics

In line with the second case-related research question, the following section describes which metrics the case organization uses so far, how it calculates, the calculation frequency, and the perceived advantages and disadvantages.

IT-provider

The use of metrics in this unit of analysis so far is primarily determined by the functional scope of Jira and runs mainly at the team level. Some of the interviewees also use Excel for individual calculations. The basis for almost all metrics is the story point estimation at the beginning of each sprint per team, which assigns a value in story points to each requirement issue. The planned issues move into the sprint backlog, and their evolving status during a sprint feeds the calculation of the following metrics. The calculation of the metric *team velocity* results from the sum of all achieved story points of a team for a sprint. Jira calculates this metric continuously and automatically. Each team determines the planned value for the velocity of a sprint individually without being prescribed from the outside. To compensate for fluctuations and thus obtain a more meaningful planning value, the company also calculates an average velocity value for the last three sprints. The aggregated team velocities on a story point basis correlate with the so-called *change velocity*. This means that when the cross-team velocity alters, the change velocity also alters to a similar extent. The reason for this is the approximately constant ratio between the change-relevant and run-relevant story points.

Change refers to the further development of the software product by implementing additional epics that are billable. The customer has already paid for run epics. When resolving incidents, for example, the development teams only fulfill their maintenance mandate. The PPO1 differentiates between change-relevant and run-relevant story points based on the ticket type using a filter on a Jira dashboard. The change velocity is then the sum of all change-relevant story points per month at the cross-team level. In addition to the monthly calculation, the PPO1 compares the change velocity values from the previous months. The change velocity calculation serves as a starting point for the person-days per epic that can be billed. The ratio of the change velocity to the cross-team velocity per epic with the average hourly rate and the time booking per epic result in the further development costs per epic. The change velocity is not communicated internally to the agile teams, as it is irrelevant for the developers whether they are providing maintenance or further development. The team velocity is sufficient. Therefore, there is no predefined ratio between run and change. The *burn-down chart* is linked to the calculation of team velocity. In this case, the metric visualizes the progress of a team sprint using a graphical representation. The estimated effort "burns down" throughout the sprint, depending on the number of completed topics. At any point in time during the sprint, this allows us to see how far the development team has progressed in completing user stories. This means full transparency about the work of the development teams and can become a demotivating factor, especially for inexperienced agile teams, if sensitive points are pursued with vigor. In the case of a *burn-up chart*, representation runs in the opposite direction, but the principle remains the same. For both, the target is to reach the ideal line, even if, according to the interviewee, this is not always possible due to the technical constraints in terms of granularity in Jira. Besides team velocity and the burn-down chart, the development teams also use the *sprint report*. It continuously calculates the amount of work in the various stages measured in story points. The sprint report includes issues that were completed or not completed during the sprint and those that were added, put on hold, or removed. A list visualizes the individual issues with their attributes such as identifier, description, type, priority, status, and story points. Additionally, Jira depicts a mini burn-down at the top of the page. The information for the metric also stems from the sprint backlog of a team of the current sprint, which can contain user stories as well as bugs or other ticket types. Overall, the metric provides a sound basis for discussion and is consistent with the preference for numbers among the predominantly mathematically trained workforce. Additional metrics exist, such as the *cumulative flow diagram* and the *version report*, which are currently not actively used, although Jira collects the metrics. On the one hand, this has to do with the fact that the metrics are not very meaningful for the case organization and, on the other hand, that the stakeholders have not yet dealt with them. Service Management also captures incidents across the organization using six different categories, which the IT Service Owner (PPO1) adds to sprints as bugs, tasks, or other Jira software issue types. A Jira Board makes the open incidents visible. A complete overview of the existing metrics is depicted in Table 4.4.

Metric	Calculation	Org. level
Change velocity	Sum of story points of all completed user stories per month that are related to change by all teams	Cross-team
Team velocity	Sum of the story points of all user stories completed by a team per sprint	Agile teams
Burn-down chart	Sum of the story points of all open user stories in a sprint of a team in relation to the time left in the sprint reduced for each completed user story	
Burn-up chart	Sum of the story points of all completed user stories in a sprint of a team in relation to the time left in the sprint	
Sprint report	Sum of story points per category (issues completed, not completed, added, removed, on hold)	
Cumulative flow diagram	Number of the work items in each status (e.g. open, in progress, done) in relation to the time per sprint	
Version report	Shows team progress towards the completion of a version with the predicted release date, based the average rate of progress (velocity) since the start of the version, and the estimated amount of work remaining	

Table 4.4.: Overview of the IT provider's existing metrics

Sales Agile Release Train

The Sales Agile Release Train has been moderate in introducing metrics compared to other ARTs within the case organization. The metrics customer satisfaction, employee satisfaction, and effort allocation are part of the group-wide standard. The case organization measures both *customer satisfaction* and *employee satisfaction* via surveys in which respondents provide qualitative and quantitative feedback and express their satisfaction in different dimensions on a scale from one to five. The organization calculates both metrics for every PI by dividing the number of top two boxes through the total number of responses. For example, in the case of customer satisfaction, there are the dimensions of satisfaction with steering roles (RTE, Product Manager, System Architect, Primary Manager), with PO and team communication, with program management communication and collaboration, and with PI preparation and planning. Alongside these three standard metrics, the teams in the ART essentially use the metrics: run & change, commitment quote, and error tickets. The *run & change* metric specifies the ratio between maintenance and new development. The calculation rule is derived by dividing the time spent for maintenance over three months or one sprint by the total time spent. The calculation is first done at the team level and then aggregated at the cross-team level. A booking tool collects the metric data when employees continuously enter their time for the run or change position. Hence, the employees themselves are responsible for entering the times truthfully, which can lead to abuse and puts pressure on employees. After three months at the end of the PI, the teams submit the numbers to the RTE. The teams strive not to exceed a value of 20 percent for maintenance to spend enough time on new and further developments. At the moment, the value is significantly higher. Therefore, the team plans intermediate steps to reduce the value. The advantage of the metric lies in the critical examination of maintenance. The metric supports the review of the current technical status of software based on technical innovations to uncover automation potentials. The second key metric, *commitment quota*, measures the proportion of planned and delivered person-days during a PI. At the beginning of a PI, the ART considers the available affordability by subtracting the maintenance effort and SAFe meeting overhead from the change capacity. On this basis, the ART distributes the resources among the different customer projects, determines how many person-days the ART can commit to the customers and how many stories this corresponds to. The ART reviews the progress every three weeks and presents it to the customer. After the PI, the ART checks how many promised stories have actually been delivered and compares this actual value to the planned value. The ART aims for a value of more than 95 percent. This translates into five percent tolerance, whereby implemented innovations can also increase the realized commitment quota. Jira calculates the metric automatically and with little effort. Various boards provide a quick overview of the status of the completed stories with a traffic light label. The interview partners consider the advantages of the metric to be early mutual support in the event of problems and the ability to react to delays or changes at an early stage. They consider conservative estimates by the teams, which should always meet the planned value, to be disadvantageous. The third of the essential metrics refers to the *error tickets* of the entire ART. The error tickets are partitioned into five categories, three of which are relevant. The tickets are created in Jira either by direct customer input or through customer support. Monthly, the ART evaluates the trend in the number of errors per category. In addition to the error tickets, the ART measures the time it takes to fix an error and the time it takes to react to the error: *error ticket fixing time* and *error ticket response time*. The overall benefits are strongly linked to the continuous monitoring of error tickets. The earlier errors are detected and handled, the higher the customer acceptance. Unfortunately, the error tickets cannot be predicted, which can affect the commitment quota. In addition to the metrics mentioned above, the ART also measures *test coverage*, and specialized testers report downtime and test success statistics. Furthermore, some agile teams use the automatic metrics provided by Jira, such as the *burn-down chart* on a sprint basis. The ART's steering board looks at the *throughput of features and stories* per PI at the program level. The measurement within the different teams is supported by various programs such as Jira, SonarQube, HP ALM, Tosca, Findbugs, Teamscale, Dynatrace, Maritz, Excel, and Xray. Table 4.5 provides an overview of all metrics used in the ART.

4.5. Distribution of responsibilities for metrics and goals

The following section provides an overview of the responsibilities and tasks of the various stakeholders in defining, selecting, using, and monitoring metrics and goals.

Metric	Calculation	Org. level
Customer satisfaction	Number of customer satisfaction ratings of the two best values of the rating scale to assess communication, collaboration, and PI with different ART-stakeholders divided by the total number of satisfaction ratings made by all customers of the ART	Agile release train
Employee satisfaction	Percentage of the employee satisfaction of the two best values of the rating scale in relation to the total number of satisfaction ratings made by all employees in the ART	
Effort allocation	Number of person days spent on each project by all teams within the ART per PI	
Commitment quota	Percentage of the person days estimated for all user stories that have been implemented within an ART per PI in relation to the person days estimated for the total number of planned user stories	
Error tickets	Number of error tickets in the production environments of the software product for each severity category	
Error ticket fixing time	Duration (end time minus start time) between the creation of an error ticket in production until the error is resolved	
Error ticket response time	Duration (end time minus start time) between the creation of an error ticket in production until a response	
Test coverage	Percentage of the number of lines of code all tests currently execute in relation to the total number of lines of code of the software that is tested	
Throughput of features/stories	Amount of features/user stories realized within a PI by the ART	
Run & change per PI	Percentage of the person hours spent on user stories for development of new features (change) or on maintenance features (run) of the software product in relation to the total effort spent on user stories of all types of features all teams of the ART per PI	
Run & change per sprint	Percentage of the person hours spent on work on the development of new features or spent on maintenance work regarding the software product in relation to the total effort spent on work by a team in a sprint	Agile teams
Sprint burndown chart	Sum of the story points of all open user stories in a sprint of a team in relation to the time left in the sprint reduced for each completed user story	

Table 4.5.: Overview of the ART's existing metrics

IT-provider

When a new team is formed, the Scrum Master takes on introducing metrics to provide transparency and understanding about development activities. The Scrum Master also takes care of the adoption of metrics since metrics are part of a framework and the Scrum Master is responsible for the adherence to the framework. The task is to explain the metrics and their purpose and, thus, ensure understanding among the development team. In the case that metrics should be abolished again, however, the consensus in the development team decides and not the Scrum Master alone. This approach applies not only to metrics but to all important decisions that directly affect the team. The team monitors the metrics in each sprint planning and commits to a specific sprint scope depending on the metrics. The development team takes full responsibility for the sprint scope. The Scrum Master exercises a supporting responsibility in monitoring metrics and monitors depending on the maturity level of the development team. The Scrum Master monitors the metrics several times a week in the early stages and reduces the workload as the maturity level progresses. On average, the metrics are monitored once a week. Especially the monitoring of the achieved scope in the second week of a sprint is important because, from this, it can be deduced to what extent the delivery scope is achieved, the schedule is reached, and which measures have to be taken in the last week of the sprint. The Scrum Master intervenes if the development team permanently does not achieve the target commitment level or does not meet their responsibility. Apart from that, the responsibility for achieving the sprint goal and monitoring it via metrics lies with the development team, and the Scrum Master supports the team in doing so. The Product Owner also is interested in monitoring metrics such as burn-down or team velocity because this allows the PO to estimate how many story points can be expected from the development team during the next sprint. The Release Manager is responsible for the release goals and metrics. The Release Manager monitors the metrics related to the release at least twice a week. Together with the Product Owner, an exchange about the progress of the release also takes place once a week. When introducing the change velocity, the IT Service Owner worked cross-functionally with the Scrum Master, Business Analysts, and Chief Technology Officer in an iterative process to find a suitable solution. Nevertheless, the responsibility for the metric lies with the IT Service Owner, as he is responsible for budget planning and capacity planning.

If there is a deterioration of the agile team metrics, this is discussed collaboratively in the Sprint Retrospective between Scrum Master, Product Owner, and the development team. The intention is to identify the reasons and to find improvement measures to solve the problems. In case of deviations from the sprint targets, the unachieved user stories become part of the backlog of the next sprint and, due to their higher priority, cause a change in the previous sprint planning.

According to CTO1, the definition of strategic goals falls into the hands of the management. The management members discuss the goals in the Executive Meta Scrum Board and then communicate them down the hierarchy. From the perspective of the ISO, the Service Owner and the Product Owner are primarily responsible for compliance with the goals around the product. The Product Owner controls the adherence to time agreements, and the Service

Owner is more concerned with the quality of the product. For security and compliance goals, the Information Security Officer shares responsibility with the Service Owner. According to PPO1, responsibility for customer satisfaction is shared by all employees because everyone contributes to making the customer satisfied with their work.

Sales Agile Release Train

The main focus of the Sales ART is to achieve openness and collaboration. Moreover, RTE1 understands metrics as part of SAFe and needs reliable numbers for decision-making. The focus here is not on control but joint further development as ART. At the beginning of the metrics introduction process, there was an agreement between the management and the agile teams in joint meetings on the goals and metrics. Hence, the teams were strongly involved in the definition of metrics or, as in the case of error tickets, they directly initiated the introduction and set the quality standards independently. RTE1 has taken more of a moderating role in the exchanges of the steering board with the agile teams. The execution of the metrics is mainly through the actions of the agile teams, booking times, maintaining tickets, or committing to user stories. Even though management sets the target values for the metrics, the teams are responsible for the planned values. Responsibility and frequency of monitoring depend on the metric. Monitoring the run & change metric is the joint responsibility of management, RTE, and agile teams following the three-month PI cycle. The agile teams monitor the error tickets metric primarily themselves according to the Service-Level-Agreements (SLA)s. There is also the support function for the teams as a shared service that tracks and controls the metric values monthly. In the case of the commitment quota, the agile teams report the metric to the RTE, who monitors it every three weeks. The potential decision to stop using the metrics must ultimately be made by management, although the teams would be involved in the phase-out. In the case of most metrics, the clients and end-users have a vested interest in the performance evolution of the software. In addition to RTE, management, agile teams, the PO of each team, and product management are also among the interested stakeholders within the ART.

Responsibility for goal definition is shared among the various stakeholders in the ART. At the team level, PM1 sees the responsibility for the goal definition with the agile teams within the framework of the self-autonomy of agile working. If gaps arise due to inadequate goal definition, the steering board intervenes by setting concrete goals and checking them using metrics. The System Architect takes over the goal definition for more technical topics. The teams are responsible for achieving goals such as development quality, transparency, and customer satisfaction. SM3 regards the Scrum Masters as more responsible for promoting communication and collaboration in the agile teams. The responsibility for the commitment goal mainly distributes at the team level between the product owner and the team (SM3).

The agile teams are responsible for deviations from the predefined target values. Possible actions for deviations in the target values include asking critical questions that address the root cause. In addition, the deviation should be communicated and reported as early as possible to find out the root cause and prevent financial and immaterial damage. For this purpose, regular synchronization meetings are held to address problems, obstacles, and deviations. Together, the participants then discuss and decide how to respond. The product manager serves as the technical contact person. Issues that are neither functional nor technical are escalated to the RTE.

4.6. Challenges in the introduction and use of metrics

This section addresses the challenges associated with the introduction and use of metrics and their countermeasures.

IT-provider

A key challenge in using metrics within the development unit of the IT-provider is to minimize the sense of control among developers. Especially in the metric adoption phase, it is essential to build trust and understand that metrics intend to improve development and related activities. The challenge amplifies against the backdrop of the framework's restrictive specifications, which impose a structure and limit freedoms. In particular, employees new to agile work find it difficult to discard the idea of control that prevails in traditional organizational forms. However, the feeling of control does not apply to all metrics to the same extent. For example, the challenge is seen as less critical for the sprint report than for the burn-down chart. As a countermeasure, the Scrum Master utilizes sensitive communication to convince the development team that it is not about control but improvement. The intent is to work toward changing the viewpoint of the development team, which ideally perceives the metric as a motivator that helps them see progress and celebrate successes together. Therefore SM1 suggests using the burn-down chart only in extreme situations with fluctuations in the metric data. The idea is to talk purposefully about the problems and discuss possible solutions without building up pressure or creating a familiarization effect.

The PO1 does not see any disadvantage in using metrics per se because they provide additional information. The challenge for PO1 and ISO1 lies in the potential misinterpretation of metrics. Based on the values of a metric, the metrics stakeholder can make statements or deductions that do not consider the influencing factors, such as the available personnel capacity in the case of velocity. A divergent understanding of the metrics can also reinforce this. The consideration of influencing factors and other metrics can counteract this. Documentation that promotes uniform interpretation and identifies the influencing factors can also be helpful.

Another challenge the organization faces in a broader sense with metrics arises when the sprint scope is increased by subsequently added tickets. This reduces the quality of sprint planning and should not be the standard. Even experienced teams can overestimate themselves. Explicitly the burn-down chart and the sprint report help to make subsequently added tickets visible. The SM1 reports that it is essential that the team "feels the pain" that this behavior triggers to prevent it in the future.

Sales Agile Release Train

Similar to the interviewees' statements at the IT-provider, avoiding the sense of control is also a central challenge in the Sales ART (RTE1, SM2, SM3). The metrics were introduced together with the agile teams to maintain the feeling of freedom. This collaboration helps prevent manipulating metrics, as might be the case when booking the run and change positions. Moreover, to achieve a change ratio of 80 percent, it is crucial on the part of the RTE and the PO not to prioritize out work packages that relate to change and at the same time to demand the ratio. For this reason, there is a certain amount of free space to implement financially sound automation projects. For PM1, a significant challenge in introducing and using metrics is not to limit personal responsibility and team autonomy. The teams should actively demand metrics to avoid the impression that the steering board is imposing requirements. Another challenge arises from the cross-sectional function of the ART, which spans all divisions of the case organization. This makes planning more challenging and involves dependencies whose effects are difficult to forecast. Countermeasures encompass, on the one hand, the possibilities of clearly assigned responsibilities in the service pyramid to which problems can be escalated. On the other hand, the ART relies on many tests to detect errors from interdependencies at an early stage. A minor challenge related to metrics, in general, was identified as the lack of ability of metrics to prioritize actions. In addition, metrics can increase the pressure to perform. However, metrics and the achievement of their target values can also be a limiting factor in the drive for continuous improvement, especially if the target values are not increased after repeated achievement. This phenomenon is called conservative estimates by RTE1. The approach to counter this is to reduce the uncertainties and risks and raise the quality of the requirements. It also helps to develop a shared understanding within communities of practices. SM2 states that the uneven use of metrics without a systematic approach leads to lower effectiveness in the targeted implementation of requirements. Also, the lack of skill-set, too many specifications, and overwhelm caused by the novelty of agile transformation can lead to an unwillingness and rejection of metrics. Therefore, it seems indispensable to transparently explain the sense and benefit of metrics, promote openness, and create the necessary conditions in the course of the agile transformation. Common standards regarding the agile frameworks, currency basis (story points or person-days), and metrics also seem to be very important (SM2).

4.7. Recommendations for the adoption of metrics

Before adopting the metric, interviewees were asked about their perceptions of successful metrics adoption in agile teams. This section describes the resulting findings as recommendations and summarizes them in Table 4.6. The recommendations are revisited and incorporated into the approach in Chapter 5.

One word that came up frequently in the interviews in connection with the adoption of metrics was transparency. SM2, SM3, and CTO1 emphasize the significance of transparency in the communication about what is necessary and what benefits the introduction offers. For this, SM3 recommends that the people responsible for the introduction engage with

ID	Recommendations
R1	Shared transparent dialogue between metrics team and agile teams about benefits and purpose of metrics adoption
R2	Convince all relevant stakeholders and promote acceptance
R3	Ensure the goal-based selection of metrics
R4	Use hypotheses and questions in addition to goals to operationalize metrics
R5	Avoid the impression of control and emphasize the intrinsic value of metrics
R6	View the metric introduction as an agile experiment to try out potential improvements
R7	Regular review and iterative adjustment of metrics, including realignment to goals
R8	Establish the basics of agile working, train employees, and provide appropriate metrics tools
R9	Prioritize the use of existing data to minimize effort
R10	The benefit of a metric must justify the effort
R11	Focus on a few meaningful metrics
R12	Assign responsibility for metrics to a person who centrally coordinates the adoption project
R13	Document metrics to ensure traceability and consistent interpretation
R14	Create awareness of the meaning of metrics and a shared understanding of the associated assumptions, limitations, and influencing factors
R15	Ensure easy accessibility and good visibility of metrics
R16	Use metrics actively in short cross-team meetings
R17	Use the same metrics throughout the same organizational level
R18	Define metric thresholds for triggering actions and develop scenarios that implement them

Table 4.6.: Recommendations mentioned during the case study

the agile teams. Furthermore, the purpose of metrics to improve development should be explained to the agile teams, and open communication should be encouraged in the exchange (R1). This recommendation is related to the acceptance called for by RTE1, Dev1, and CTO1 because the organization can achieve acceptance through transparency in communication and education, among other things. It is vital to convincing all stakeholders, those who are being measured and those responsible for the measurement (R2). According to SM3, aligning metrics with goals can also contribute to acceptance. In principle, the selection of metrics based on goals is indispensable, as confirmed by CTO1, Dev1, and SM3. If this is not the case, there is also the danger that the agile teams will optimize the metrics to correspond to desired values because they do not see sense in the metric (R3). Metrics serve as indicators of goal achievement. Hypotheses and questions can be applied to steer the organization in the right direction (RTE1). Therefore, hypotheses and questions should be part of the introduction of metrics in addition to goals (R4). The acceptance, however, could be undermined by the perception of the agile teams that management wants to exert control employing metrics (SM2, SM3, Dev1). Therefore, it is essential to avoid this perception and to emphasize the intrinsic benefits (R5). Some interview participants (SM2, SM3) suggested considering the introduction of metrics in line with the agile principle as an experiment and trying out the use with an open result (R6). According to CTO1, this also includes regular review and adjustment of metrics (R7). Just as regularly, the organization should review goal attainment after adoption (Dev1). SM2 regards the establishment and understanding of agile working as a prerequisite for the introduction of metrics. Otherwise, the introduction of metrics can overwhelm agile teams. This establishment includes training on using tools such as Jira (ISO1) to support agile working and automatically capture metrics (R8). When introducing metrics, CTO1 recommends using existing data first before collecting new data with additional effort. This stance is the starting point even though improvement through new data sources must continuously be re-evaluated (R9). The organization should weigh the benefits against the effort required to introduce a metric (R10). This weighing also has a lot to do with the focus on essential metrics mentioned by ISO1. Instead of introducing many metrics and measuring a lot, ISO1 recommends introducing a few metrics with particular significance and less costly data collection (R11). In the view of ISO1, the introduction of metrics should be coordinated in a centralized manner by one responsible person. This person ensures that the introduction fits into an overall concept and that there are no duplications of metrics. In addition, this person acts as a contact person for new metrics projects and provides clarification (RTE1) (R12). The person also takes on another recommendation (PO1, ISO1), the uniform documentation of metrics (R13). By describing the metrics, the documentation creates a common understanding of how to interpret them and provides traceability of their adoption. Echoing this recommendation, CTO1 and ISO1 recommend creating awareness of the meaning of metrics and developing a common understanding of the associated assumptions, limitations, and influencing factors (R14). This awareness is especially relevant for predefined metrics from Jira, for example, whose meaning is too rarely critically questioned. An example of an influencing factor of a metric is the correlation between overtime and velocity, which should be considered when using it. In general, there are a couple of things to consider after introduction for sustainable and long-term adoption of metrics. First, visibility and accessibility of metrics are very crucial according to Dev1, PO1, CTO1. These two requirements allow for more frequent use, transparency, and results-based discussion (R15). Second, the regular, intentional use of metrics in short cross-team meetings contributes to sustained metrics adoption (Dev1) (R16). Third, especially in a scaled agile environment, it is essential for adoption that metrics are introduced and used consistently at an organizational level (R17). Otherwise, agile teams will have a harder time implementing the requirements effectively and goal-oriented (SM2). Finally, from PO1's point of view, it makes sense to define thresholds for actions and develop scenarios linked to the metrics and trigger certain interventions. For this purpose, organizations can build up a pool of experience over the years of use and document it (R18).

5. Approach to introduce metrics and improve measurement program management

This chapter presents the generalization of the comprehensive solution artifact in the context of this thesis: an approach for goal-based metrics introduction and improved management of measurement programs in large-scale agile software development. Development organizations in almost every company have established measurement by metrics following the agile principle of measuring progress against a working product. The so-called measurement programs build on the elements of traditional measurement and increasingly extend or replace these with agile metrics and approaches. In large-scale agile environments, frameworks are also integrating project and portfolio management structures. The approach formulated in this chapter aims to support development organizations in this context by introducing new metrics based on their goals and, optionally, by professionalizing the metrics team as a whole. The approach builds on insights from software measurement research and combines several components. The specific problems that the approach solves as an artifact in terms of the action design science method [102] are primarily the lack of alignment between metrics and goals [47, 83], the difficulty of deriving actionable value from metrics [90], and lack of skills and guidance in introducing metrics [58]. To date, there is no scientifically sound reference for the systematic introduction of metrics involving all relevant stakeholders in the context of large-scale agile software development. Besides the practice-inspired design of the artifact resulting from the case study collaboration, various success factors are embedded into the steps of the approach and, thus, contribute to the theory-ingrained foundation. The final part of the chapter revisits the success factors and summarizes them together with the recommendations from the case study in an overview that supports the approach. The approach ties in with the proven four phases of the GQM-method and supplements them with an initial analysis of goal coverage through the metrics used so far and a periodic review of goal coverage after adopting new metrics as part of the agile meetings. Furthermore, the approach serves the goal-based introduction of metrics in the large-scale agile context and offers the flexibility to optionally improve the measurement program with its associated measurement team in a holistic sense. The approach also grants the introduction independent of the already existing agile framework.

The chapter consists of three parts. Section 5.1 describes the components on which the approach relies. These include the involved stakeholders, the metrics catalog, the GQM-method, and the MeTeaM model. Section 5.2 then explains the process of the approach and its steps. Finally, Section 5.3 summarizes all recommendations that were shared with the metrics teams from the case study.

5.1. Components of the approach

This section describes the various components that shape the approach. One essential component is the group of stakeholders to whom the approach is directed. Also indispensable is the metrics catalog that provides the data foundation for the goal-based selection of metrics. Brief descriptions of the GQM-method and the MeTeaM model follow, which are already familiar from subsections 2.3.5 and 2.3.6.

Involved stakeholders and their role in the approach

The approach recommends to involve any employees responsible for planning, defining, maintaining, operating, and managing metrics to adopt the approach. The employees can work in a dedicated and formally defined measurement team or come together in a virtual team where the members work distributed in many different teams and roles [105]. The latter seems to be much more common in practice, especially in smaller organizations. Generally, it is apparent that primarily Product Managers, Scrum Masters, Product Owners, Release Train Engineers, and developers perform metric-related tasks in large-scale agile organizations. The division of responsibility varies from company to company, but it has proven successful in involving all affected stakeholders in the metric introduction process [17, 104]. The goal formulation and the subsequent selection of metrics should ideally be set up around these individuals and teams [34, 47]. In addition, the successful metrics introduction is highly dependent on the commitment of management and teams, which needs to be ensured [104]. Overall, it is practical to designate one person for the role of leading the introduction who is responsible for most of the administrative issues of the team. This person can be the existing metrics team leader, as defined by Staron and Meding [105], or another person if the organization does not have formally defined metrics team roles. Furthermore, the approach envisions that a person or a group is accountable for planning, executing, and evaluating the metrics introduction and its success, similar to measurement analysts. Two other individuals or groups of individuals manage the measurement data and are responsible for the measurement process. This division of responsibilities and roles ensures the successful implementation of the approach and should be clarified in advance.

Metrics catalogue

The metrics catalog is the data basis for mapping the goals with the metrics. The mapping works both ways. When analyzing the existing metrics, the catalog serves as a reference for the goals covered so far. For the goal-based selection of new metrics, once again, the catalog helps by its offered goal-metric matching. The catalog consists of 196 metrics that were collected using interviews with 20 participants from over ten companies. The metrics are defined either at the team, program, portfolio, or organizational level. Each metric maps to one or more goals from the five major categories identified. The five main categories divide into 27 subcategories of goals. The Goal-Metric-Matrix allows navigation through the catalog. A page of the metrics catalog generally consists of title, description, code, organization level, goals, information model, and calculation formula [111].

GQM-method

Goal orientation is one of the success factors in metrics adoption because it increases awareness and dedication of all involved stakeholders [71, 91]. The Goal Question Metric method is the starting point for goal-based metrics adoption. Contrary to bottom-up approaches, the GQM-method relies on the inclusion of the context in the definition of metrics, which is why metrics are selected top-down based on goals and then interpreted bottom-up. The advantage is that the alignment with organization-specific goals ensures that metrics are selected in a way relevant to goal achievement [59]. The GQM-method includes four distinct phases, as illustrated and described in Figure 2.5 of Subsection 2.3.5. In the initial planning phase, the GQM team gathers, selects the improvement area, and develops a project plan that characterizes the process to be measured and establishes the measurement schedule. The goals, questions, and metrics are defined in the subsequent definition phase, which results in a GQM plan, a measurement plan, and an analysis plan. Measurement by the selected metrics takes place in the data collection phase. The final interpretation phase verifies that the measured metrics have met the goals and answered the questions [59]. These four phases are also reflected in the procedure of the approach. Besides, the suitability of the GQM-method in the context of agile development could be proven by several studies (see Section 3.1).

MeTeaM-model

The quality and effectiveness of measurement programs depend not only on the selection of metrics and their use but also on the maturity of metrics teams, as Meding et al. conclude [73]. Using the assessment model they present, it is possible to characterize a metrics team to uncover the improvement potential along several dimensions. Since metrics teams design, deploy and manage measurement programs, organizations should understand how to best support the teams in doing so. The MeTeaM model builds on observations of the needs of software metrics teams in practice and the findings of research on team maturity and software measurement programs. These observations and findings translate into the six areas of the model: set-up, commitment, impact, longevity, efficiency, and team performance. Subsection 2.3.6 provides a detailed description of each area. The model assesses metrics teams based on these areas using a template and visualizes the result so that steps can be taken to make improvements. The model serves as another component of the approach and complements the goal-based implementation of metrics. Agile metrics teams get the chance to professionalize their measurement actions in a holistic, systematic, science-based, and flexible way. The model and its template are applied in the second part of the approach in 5.2.2.

5.2. Procedure of the approach

This section explains the procedure of the approach for the two parts goal-based metrics introduction and measurement program improvement. Both parts can be performed independently of each other. The approach offers flexibility for the organization to decide which part to use or whether to use both.

5.2.1. Goal-based introduction of metrics

The procedure for goal-based metrics introduction consists of eight sequential steps, with the eighth step to be performed regularly. The eighth step can initiate a repetition of the entire process, making the approach iterative. This characteristic of the approach is a proven success factor for metrics adoption. Various authors recommend developing the measurement program over time and introducing a few key metrics at the beginning. It is essential to wait for the agile organization to adapt and not trigger an overload [42, 83, 91, 104]. After a successful introduction, the approach envisages reviewing the need for additional metrics and, if necessary, restarts the overall procedure. This procedure is similar to the Plan-Do-Check-Adjust cycle as performed by SAFe in its iterations. The aim is to create faster learning through faster cycles, according to the fourth principle of SAFe. The approach is about experimentally finding out [R6] and learning which metrics help the organization achieve its goals. When implementing the approach, there are also some general factors to consider to introduce metrics successfully. Ideally, the organization will deploy the same metrics across all teams at the same level, ensuring horizontal alignment [R17] [104]. Additionally, it is also paramount to define metrics at all organizational levels and align them with the goals at the different levels [34, 83]. On top of that, the synchronization cycles of all organizational levels in the organization need to be aligned [47]. The alignment in terms of time and content ensures that decisions are made on the same basis [89]. Another success factor in the introduction of metrics is the commitment of management and the team. Management commitment is expressed financially through the provision of tools and personnel and the support in defining goals. The team commitment is necessary for ensuring the reliability of the measurements and the continuous improvement of the ways of working [104]. The aspect of engagement reappears in the second part of the approach within the maturity assessment. Figure 5.1 provides a general overview of all steps of the goal-based metrics introduction procedure. The following paragraphs explain the procedure of the first part of the approach in detail, step by step.

Step 1: Identify existing metrics

In the first step, the metrics team identifies the existing metrics by analyzing the metrics tools or having joint discussions. The identified metrics should be collected and recorded in a list, including the description, organizational level, calculation formula, calculation frequency, and measurement unit. The step is led by the leader of the introduction, who also takes care of the documentation. The collected metrics are then further utilized in the next step. In the case that no metrics exist, this step is omitted.

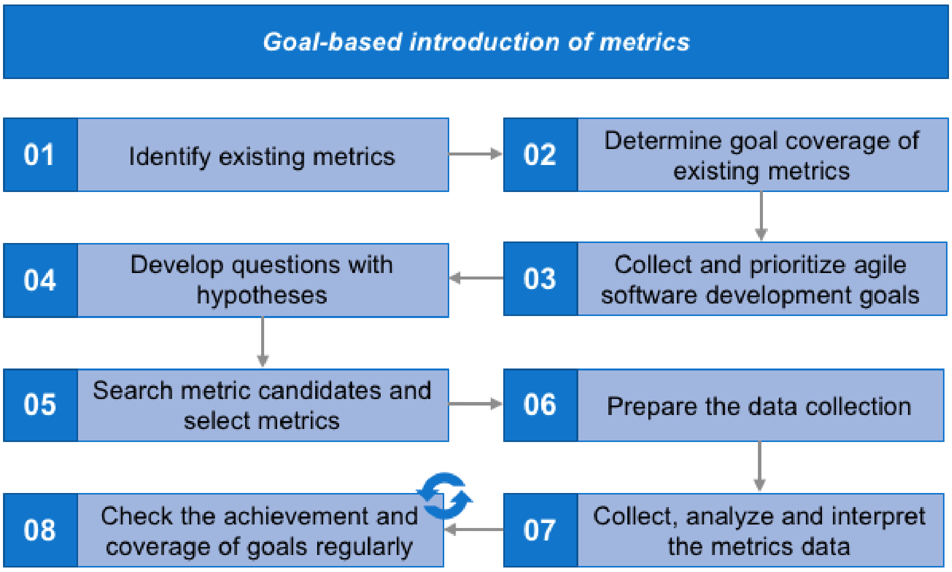

Figure 5.1.: Procedure of the goal-based introduction of metrics

Step 2: Determine the goal coverage of the existing metrics

In the second step, the metrics catalog is used for the first time. Using the Goal Metric Matrix, the metrics team selects the goals at the subcategory level for each collected metrics. If there are goals for a metric that the catalog does not contain, the team has the flexibility to add those goals. The introduction leader documents the fulfilled subcategory-level goals in a list and calculates the goal coverage as a percentage. The leader also sorts the goals in the list in descending order by goal coverage percentage. The goal coverage serves the metrics team as an indicator for the subsequent prioritization of the open goals in the next step.

Step 3: Collect and prioritize agile software development goals

Step 3 starts with the collection of agile software development goals at the conceptual level [8]. For this purpose, the metrics team should interview representatives of all roles at all levels in the agile development organization. This group of persons may include Scrum Masters, Product Owners, Developers, Test Managers, System Architects, Product Managers, Release Train Engineers, and managers. The multitude of roles at different levels creates a comprehensive picture of goals, which should be aligned with the goals of the Agile Manifesto and the organization's overall business goals [47]. The approach empowers teams to participate and actively influence goal definition through their representatives. Self-defined goals improve the team's understanding of the purpose [47]. However, the representatives must communicate the results to the team on their initiative. In smaller development organizations, it can certainly be beneficial to involve all team members directly. In larger organizations, this is hardly possible in a reasonable time frame. The interview questions include the aspects: software development goals, goals not covered by metrics, responsibilities related to goals, and priority, purpose, problem, object, perspective, context, and questions for each stated goal. With this information, the metrics team can prioritize the goals and document them in a structured way according to the GQM-template. The prioritization considers the information from the interviews and the list of goals, and their coverage percentages. Both sources help narrow down the number of goals. Limiting the number to a maximum of the top three goals is advisable to avoid making the metrics candidate list in step 5 too large. The metrics team can count mentions of the same goals from the interviews and exclude less important ones with few mentions. Similarly, goals with low coverage and frequent mentions can be prioritized over goals already very well covered.

The documentation requires that the introduction leader prepares a GQM-plan for each of the goals. Each goal has a purpose (e.g., improve, increase, achieve) that addresses a specific issue (e.g., transparency, quality, commitment). The issue relates to an object (e.g., features, requirements, capabilities) from a particular perspective (e.g., customer or agile teams) in a context (e.g., ART's agile development environment). A concrete example of a goal documented using the template could be:

> "Improve (=purpose) the quality (=issue) of software product X (=object) from the viewpoint of customer Y (=perspective) within development department Z (=context)."

The metrics team openly communicates all goals and their priority to the agile teams after the interviews. Based on all the information, the metrics team should then decide which goals to select and introduce through metrics as part of this iteration. The metrics team documents excluded goals and the interview results so that upcoming iterations can incorporate the information. The started GQM-plan continues in the next step.

Step 4: Develop questions with hypotheses

In Step 4, the GQM-plan table fills up with the development of questions. The questions are used to refine and operationalize the abstract and conceptual goals [8]. They line up at the level between the goals and metrics. For each goal comes at least one question, usually several. A question expresses the informational interest of the metrics team. By answering the question, the team can judge whether the goal is being met. The level of detail of the questions should ideally have a recognizable relationship to the data and allow interpretation concerning the goal. For each question, the metrics team can develop a hypothesis according to the GQM method. This hypothesis captures the expected result and thus allows to compare the knowledge before and after the measurement. As a result, the metrics team can increase the learning effect. For example, a question might look like this:

> "How many defect tickets does Team X produce during a sprint?"

A corresponding hypothesis could be as follows:

> "Team X produces Y defect tickets during a sprint."

Step 5: Search metrics candidates and select metrics

Once the conceptual-level goals and operational-level questions are established, the metrics team can compile a list of metrics candidates at the quantitative level [8]. The Goal-Metric-Matrix of the metrics catalog and the framework in use help in the goal-based search for metrics. The Goal-Metric-Matrix filters appropriate metrics based on the identified goals, which become part of the candidate list. Level two of the goals serves as the entry point for searching for candidate metrics within the catalog. The catalog does not cover all imaginable goals. If a goal mentioned cannot be found in the catalog, it is then at the discretion of the metrics team to select similar goals as proxies of the actual goal. Some frameworks provide metrics that can also be included in the selection step. Metrics that the agile teams previously calculated manually can also be included in the candidate list to automate them. The approach gives the metrics team the freedom to adapt or completely redefine metrics from the above sources. However, the candidate list should not be too long to prevent the metrics team from losing track. In principle, a metric can help answer different questions and confirm or refute multiple hypotheses even if they belong to different goals. The selection of metrics should be done in coordination with the metrics team and set up around agile teams and stakeholders. The hallmark of a successful measurement system is that it is linked to a stakeholder to prevent metrics from not being used after their introduction [17, 104]. In addition, it is advisable to perform a cost-benefit analysis to avoid the introduction of unhelpful metrics [104]. The metrics team should take a step-by-step approach for the introduction and limit the selection to a specific set of metrics. Only essential metrics should be selected with the potential of improving the fulfillment of the information need [83]. Therefore, if the selection of metrics is extensive, it may help divide the metrics into different term categories. An example is a three-way division into short, medium, and long-term metrics.

Step 6: Prepare the data collection

Step 6 follows the selection of metrics and creates a measurement plan that specifies how to obtain the raw data for the metrics and extract from that data the information needed for decision making. The input is the GQM-plan, which includes the goals, questions, information needs, and metrics. Note that the measurement plan contains base metrics. According to Basili et al., this refers to metrics that relate to directly measurable attributes of objects. The measurement plan does not include derived metrics. The derived metrics are calculated in the subsequent step by aggregating the base metrics, e.g., with the help of a mathematical formula. The measurement plan should answer the following six questions for each base metric [7]:

1. **What is measured? (object and attribute)** Each metric relates to a specific agile software development object and its attributes. These can be derived from the associated goal. In any case, the measurement plan should specify them in more detail. Examples of an object are the scope of sprint planning, test cases, or the backlog.

2. **Where will the measurement be performed? (Scope)** This question specifies the orga-

nizational scope in which the collection and analysis of data for each of the metrics will take place. The scope can include the entire agile development unit or just individual teams.

3. **When will the measurement be performed? (Time)** The measurement plan defines the time when the metrics team collects the data for each of the metrics. The time can be based on the calendar, an event, or the frequency of data collection. Appropriate examples include the end of the week, after each sprint, or monthly.

4. **Where is the source of information? (Source)** The answer to this question describes the location or system in which the data of each metric gets stored. The source of information concerns both the object and its attributes. In the case of the sprint planning scope, the source of information could be Jira, which tracks all story movements. In the test case example, it is SonarQube, which captures the open and completed test cases.

5. **Who is responsible for the measurement? (Responsible)** The measurement plan also calls for naming the people responsible for each metric. The responsibility can be divided into three roles: provider, collector, and validator. The provider is the source of information and provides the measurement data. The collector takes care of collecting and storing the measurement data in a specified repository. The validator is responsible for checking the trustworthiness of the data. In practice, the provider and the collector may be the same person, depending on which system collects the data. It is recommended to separate the role of the validator from the other two to ensure independent validation of the accuracy of the data.

6. **How should the measurement be conducted? (Means)** The measurement plan also specifies how each metric will be collected. Two basic approaches are available: manual and automatic data collection. In automatic data collection, a tool measures the metric after configuration without human intervention. In manual data collection, the collector gathers the measurement data, perhaps by using simple tools such as Excel. To minimize the effort in time and cost, the metrics teams should use tools that automatically collect the data and calculate metrics [42, 91, 104]. The higher and more frequent the automated data collection distinguishes, especially companies with a higher agile maturity level [108].

The preparation of data collection is the responsibility of the metrics team leader, who takes care of close coordination with the other metrics team members and the agile teams. It is essential to gain the trust of all stakeholders in the introduction of the metrics and convince them that the metrics data will not be used to their disadvantage [91]. Convincing them through positive examples of internal company users can help. In addition, the assurance that there will be no sanctions regardless of the final metric values can relieve agile teams from negative fears. The transparency factor plays a very influential role in establishing trust [83, 91]. Agile team members should understand the goal and information interest behind each metric, the calculation rule, and the user group of each metric. Visibility and easy access to the metrics can also contribute [17, 58, 83]. Making a concerted effort to use metrics in agile meetings actively also contributes to transparency. A valuable way to minimize the effort required to introduce metrics is to use existing data [91, 104]. If this possibility exists, it should be taken into account accordingly in the planning of the data collection. The same applies to the alignment with the agile development process. Integrating metrics into the agile development process using existing tools such as Jira also increases the acceptance of metrics because the learning effort is minimal or eliminated [91, 104]. Even though changing metrics for each team is considered a success factor [22], it is not advisable to use differently defined metrics in a scaled environment because this prevents aggregation at a higher level. However, coordinated and consistent deployment of customized metrics can be helpful.

Step 7: Collect, analyze and interpret the metrics data

Before data collection, the metrics team can create an analysis plan that specifies when, by whom, and how the data will be analyzed and provided to decision-makers. It also remains important to note that the data being collected should meet specific data integrity requirements. These include completeness, internal consistency, external consistency, plausibility, and the separate handling of abnormal data [7]. Ideally, the metrics team embeds the actual data collection into the existing agile development process. For example, one or more sprints can serve as a test phase for data collection. Data is then collected continuously during this phase, either by using automated measurement tools or manually by asking people. During data collection, the validator checks the data mentioned above integrity requirements. In case of violations, the validator talks to the data collectors to correct the data. If there are repeated violations, the data collection process should be reconsidered and improved [7]. The metrics team can also complete the data collection retroactively by applying the selected metrics to data from past sprints. Before analyzing the validated data, it may need to be processed and aggregated. Data aggregation involves deriving complex metric data from the base metric data using formulas. Usually, data aggregation takes place within automated tools. In rare cases, data must be aggregated manually in Excel, for example. Basic descriptive statistics such as the minimum, maximum, distribution, average values, or most frequent values can be used when analyzing the data. Appropriate visualization also plays a key role, providing quick and accurate insight into the nature of base and derived metrics. The metrics team should make sure that metrics are visible to everyone to increase transparency and trust among agile teams [17, 58]. One way to do this is to establish easily accessible dashboards that provide a centralized view of all metrics within the organizational unit [83]. Furthermore, in the context of large-scale agile software development, it is recommended to standardize reporting management systems across all organizational levels. The advantage is that all acting persons use the same information and that the data does not have to be reprocessed with additional effort [89]. The prerequisite is cross-level data integration, in which, for example, the metrics of all agile teams are displayed in Jira in an integrated manner at the program level. Finally, the metrics team interprets the collected and analyzed data. The interpretation runs bottom-up. First, the team assesses whether the measurement answered the respective questions in the GQM-plan. Second, one level up in the plan, the team evaluates the achievement of the goals considering the answered questions and the measurement.

Step 8: Check the achievement and coverage of goals regularly

Step 8 represents the regular execution of the achievement and coverage of the goals as step 7 carries it out for the first time. The metrics team should embed this iterative routine into one of their existing agile events. One of these iterative routines could be the Iteration Retrospective event on the team level or the Scrum of Scrums event on the program level in Essential SAFe. The regular review of the achievement and coverage of the goals requires the frequent monitoring of the metrics. As one of the success factors concerning the use of metrics, this allows the exploitation of the improvement potential and increases team motivation [58]. Frequent metrics monitoring should also be part of existing agile events, such as the daily Scrum meeting [17]. It allows to gain insights and make deductions to decide whether the questions and goals of the associated metric have been answered and achieved by the measurement or not. In the positive case that a metric contributes to answering the questions and, thus, achieves the associated goal, it can be retained. However, continuous monitoring is still necessary. In the negative case that the metric cannot answer the question, the metrics team should modify it or replace it with another metric. The latter would entail a new introduction. In any case, the metrics team should communicate the changes transparently to the affected development teams. In addition, the adjustment may result in a modification of the GQM-plan and measurement plan. These measurement program modifications apply in the case of unchanging goals but inappropriate metrics. In the case of changing or new goals, the metrics team needs to adapt the questions and the associated metrics. Again, the GQM-plan and measurement plan may change. Whichever the case, the GQM-plan is part of the regular review in this step. The adaptation may also mean that the metrics team drops existing metrics and adds new ones. The regular agile feedback sessions between development teams and the metrics team during and after the introduction allow for re-prioritization and adjustment of metrics and their goals and learning on both sides [91]. Especially at the program level, frequent review is essential because the importance of the metrics can change as the agile practices mature and the goals shift as well [58].

The metrics team can also consider looking at additional goals that the interviewees mentioned in the initial rollout but dropped out and introduce corresponding metrics. Besides, as the measurement program progresses, the metrics team should collect and define potential interventions that help if specific tolerance or target values of a linked metric are not met. This collection can support the organization to build a knowledge base and evolve from continuous learning [R18]. In addition to the success factors already mentioned, such as automated data collection accompanied by the least effort necessary, Ram et al. point out other factors for long-term success. These include the use of standardized base metrics that can be used to calculate various aggregated metrics. Also, it can make sense over time to establish quality criteria for metrics and measure them through indicators. Furthermore, the success of the goal-based measurement program requires the long-term dedication of the metrics team. Finally, it is worth noting that organizations should not use measurement programs to assess individuals [91].

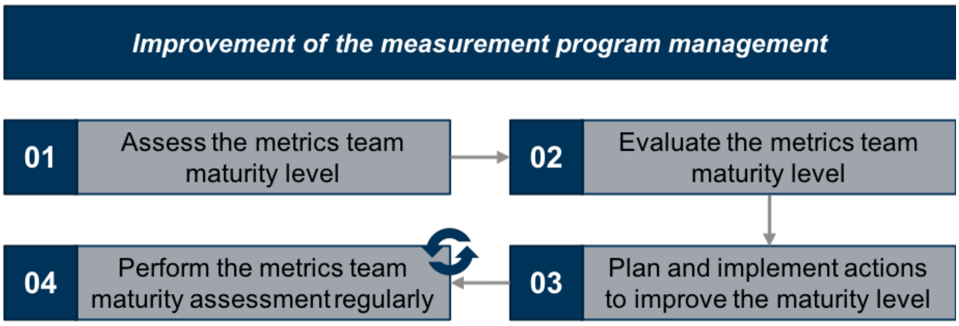

Figure 5.2.: Procedure to improve the measurement program management

5.2.2. Improvement of the management of the measurement program

This part of the approach can be considered as an add-on to the introduction of metrics. It is optional, and the metrics team can perform it flexibly, independently in time, to take actions that professionalize the team and, thus, improve the management of the measurement program. The approach consists of four steps. The fourth step expresses the iterative nature that this part of the approach also has. The visualization of the procedure is shown in Figure 5.2. The subsequent paragraphs explain the individual steps.

Step 1: Assess the metrics team maturity level

The assessment of the metrics team maturity level can be conducted in several ways. In the original paper, Meding et al.'s research group conducts interviews with members of the metrics team and its stakeholders. Alternatively, they suggest meeting participation and document study as options [73]. In addition, there is also the possibility of conducting group interviews with multiple participants or sending out the assessment template to be completed remotely. In the case of the group interview, participants agree on a response through discussion. This form is particularly suitable if the metrics team is small because otherwise, the communication and coordination effort increases rapidly. In the case of remote completion of the template by the participants, the simple rule is that the majority decides. If the answers are too divergent or distributed evenly on more than one answer choice, the participants should be asked again, or a previously determined tie-breaker should decide. However, the options that rely on the joint presence of the participants are to be preferred because consensus is easier to achieve that way. The template consists of 72 statements to which the participants can respond with four possible degrees of agreement. Fully compliant means that the metrics team adheres to the statement description in documentation and implementation as well as in usage. Partially compliant indicates incomplete conformance in the statement. This can either be caused by a lack of documentation despite implementation and usage, the other way around, or discrepancies between documentation and implementation or usage. The missing response option denotes that the metrics team does not document, implement, and use the statement's content. The fourth answer option declares that the statement is not relevant.

The statements divide into the six areas of set-up, commitment, impact, longevity, efficiency, and team performance. The set-up area has three sub-areas: formalization, roles in the team, and roles in the organization. The first sub-area is about formal recognition, tasks, resources, ways of working, and the vision/mission of the metrics team. The second and third sub-areas deal with the roles of an ideal metrics team as defined by Staron and Meding [105]. The second area, commitment, is divided into the sub-areas of team and organization. Both address commitment to standards, ways of working, and continuous improvement. The third area measures the impact the measurement program has on the organization and the outside world. The fourth area attempts to capture how valuable the measurement program is over time. In addition, the statements focus on the relationship in continuance between the organization and the metrics team. The fifth area is about how efficiently the team operates. This efficiency manifests in the provision of measurement infrastructure, speed with which new metrics are introduced, and reliable implementation of measures. The last area has several sub-areas. The statements revolve around team performance, which reveals itself in understanding context, mission, talent, norms, buy-in, resources, courage, and results. The detailed statements of the MeTeaM template can be found in Appendix A.2.

Step 2: Evaluate the metrics team maturity level

The evaluation in Step 2 follows the assessment in Step 1. The responses in each of the four levels of agreement are counted for each area and sub-area. The maturity level can then be calculated for each area and subarea. For this purpose, the metrics team multiplies the number of completely compliant statements by one and the number of partially compliant statements by a factor of 0.5. The team then sums up both results. This sum is divided by the total number of statements in the area, excluding the non-relevant statements. With these percentage values for each area, the team‘s maturity level in each of the six areas can now be visualized with a radar diagram. The radar diagram can also be drawn with two lines as an alternative to the line based on calculated percentages. The first line includes the simple percentage for fully compliant in the different areas, and the other line consists of the sum of the fully and partially compliant percentages. Either way, the visualization gives a good overview of the agreement with the statements per area. The calculated and visualized data can then be used to validate the assessment during a workshop or meeting. Participants should ideally include the entire metrics team and, if possible, representatives of stakeholders, measurement sponsors, and line managers. The aim is to develop a shared understanding of each area’s outcome and prioritize the improvement potentials constructively. It is advisable to prioritize the individual areas first and then select those on which the metrics team wants to focus. Subsequently, the metrics team also prioritizes the sub-areas, if available, and the individual statements. Based on the prioritization, the metrics team can decide which items they would like to address. Limiting the selection to one area or two sub-areas is recommended not to overwhelm the team in their improvement efforts. The metrics team leader documents the selection and records the rationale behind every selected area, sub-area, and item. Once the metrics team has reached a consensus on selecting improvement areas, it is possible to proceed to step 3.

Step 3: Plan and implement actions to improve the maturity level

Based on the selection, the metrics team plans actions to improve the maturity level in step 3. The approach recommends to proceed incrementally and try to implement one action after the other. The actions can be derived directly from the statement. In addition, Meding and Staron provide explanations and further reading on implementing the individual statements in their article and book [73, 105].

Step 4: Perform the metrics team maturity assessment regularly

To carry out the part of the overall approach regularly results from the agile principle of continuous improvement. Therefore, it is advisable to integrate the process into one of the already existing and recurring meeting artifacts of the used agile framework. The metrics team can assess the maturity level within an agile event like the Innovation and Planning Iteration in SAFe. This tri-monthly recurring event iteration allows the metrics team to perform the assessment, plan the actions, and work through them until the next iteration. A time interval of three months or more is recommended; otherwise, little change will be measurable in the maturity assessment, which would not justify the effort. However, a gap of more than one year should the metrics team avoid watching out for the metrics team's development.

5.3. Summary of recommendations for metric adoption

I developed the recommendations summarized in this section to support the adoption of metrics in the case organization. The recommendations build on the findings from the interviews (see Section 4.7) and the scientific research of Treiber [112] and Ram et al. [91]. Table 5.1 includes all recommendations with their respective sources. The division into sections denoted in italics does not follow any scientific basis but serves clarity. Since Section 4.7 and this chapter already addressed each recommendation, this section focuses only on the summary.

Recommendations	**Source**
Preconditions for the introduction of metrics	
Establish the basics of agile working and train employees	[R8]
Assign responsibility for metrics to a person who centrally coordinates the adoption project	[R12]
Features of the introduction	
Introduce metrics iteratively	[83, 91]
Ensure the goal-based introduction of metrics	[R3], [55]
Use hypotheses and questions in addition to goals to operationalize metrics	[R4]
View the metric introduction as an agile experiment to try out potential improvements	[R6]

Document the adoption of metrics	[R13]
Metrics definition and selection	
Focus on a few meaningful metrics	[R11], [83]
Verify that the benefit exceeds the effort for a metric	[R10], [104]
Allow for the coordinated and uniform customization of metrics	[22]
Data collection	
Provide accepted and appropriate tools for automated data collection and calculation of metrics	[R8], [91, 96, 104, 108]
Prioritize the use of existing data for metrics definition	[R9], [91, 104]
Collaboration and participation	
Engage in a collaborative transparent dialogue with agile teams about the benefits and purpose of adopting metrics	[R1], [17, 83, 91, 104]
Make metrics visible and easily accessible to everyone	[R15], [17, 58, 83, 96]
Involve the agile teams in defining goals and metrics	[34, 47, 96]
Convince all relevant stakeholders and promote acceptance and commitment	[R2], [104]
Ensure that the agile teams trust the measurement program	[91]
Avoid the impression of control and emphasize the intrinsic value of metrics	[R5]
Create awareness of the meaning of metrics and a common understanding of the associated assumptions, limitations, and influencing factors	[R14]
Organizational alignment	
Integrate the metrics into the flow of the agile development process	[91, 104]
Apply the same metrics at the same organizational level	[R17], [104]
Define goals and metrics at all organizational levels	[34, 47, 83]
Standardize reporting management systems across all organizational levels	[96]
Long-term and sustainable use of metrics	
Use metrics actively in short cross-team meetings	[R16]
Monitor metrics regularly	[17, 58, 104]
Define metric thresholds for triggering actions and develop scenarios that implement them	[R18], [17]
Review goals and metrics regularly and readjust their mutual alignment	[R7], [58, 91]

Table 5.1.: Summary of all recommendations for the goal-based introduction of metrics

6. Execution of the approach to introduce metrics and improve measurement program management

This chapter describes step by step the execution of the two parts of the approach as presented in chapter 5. The execution splits into two organizationally separate instantiations within the insurance group. The first is that of the IT-provider, which performs the introduction of new metrics using the approach and the assessment of the metrics team maturity level. Section 6.1 elaborates on this instantiation of the approach. The second instantiation can be found in Section 6.2, which outlines the Sales ART's progress and experience in implementing the approach. The Sales ART also executed the introduction of new metrics and the improvement of the metrics team maturity level. Together, the two distinct development organizations constitute two instantiation versions of the comprehensive solution artifact within the action design research of the thesis.

6.1. Execution of the approach at the IT-provider

Involved stakeholders and their role

The IT provider's metrics team exists virtually but is still at the beginning of its development because the entire agile development unit is under construction. First and foremost, the Scrum Master and the Product Owner deal with metrics and actively use them. Other actors such as the Chief Information Officer, the Information Security Officer, the IT Service Owner, and individual developers selectively focus on some metrics. Additionally, there is the Executive MetaScrum event in which management discusses directional decisions and goals with the Principal Product Owner and the Scrum Master. Overall, it can be stated that the Scrum Master of the agile teams is currently considered the central contact person at the IT-provider and has something like the unofficial metrics team leadership. In collaboration with this person, the approach was coordinated and implemented at the IT-provider.

6.1.1. Goal-based introduction of metrics

Step 1: Identify existing metrics

At the beginning of the approach, we identified existing metrics based on the interviews. The interviewees represented various perspectives on product, finance, security, and development through their different roles. In total, this allowed us to collect seven distinct metrics that exist in the agile development context. These metrics originate almost exclusively from Jira and belong to the predefined metrics there. The most important existing metric is team velocity, which measures the realized effort per sprint in the form of story points. Among other input factors, the team velocity helps to calculate the change velocity on a cross-team level. This metric measures the person-days spent on change story points per month and, thus, has a billing-relevant function. The burn-down and burn-up charts visualize the remaining work and the work already completed per sprint and team. These metrics are measured in user stories. Moreover, the agile teams use the sprint report to get an overview of their work. This metric provides a list with the number of story points per category such as completed, not completed, estimated, removed from the sprint, and included in the sprint. The metrics cumulative flow diagram and version report were also mentioned during the interviews, although the teams do not actively use them. The cumulative flow diagram visualizes a team's workflow during the sprint by showing the number of work items per status. The version report informs about the progress of a new software version and predicts, for example, the release date. Table 4.4 and the associated text in Section 4.4 present these metrics in more detail, including the specific calculation. From the table, it becomes evident that the agile teams primarily measure within their team boundaries and have not yet defined shared, overarching metrics. The exception remains change velocity, although this metric is not communicated to the teams. This picture reinforces the metrics team's intention to introduce more metrics at the cross-team level.

Step 2: Determine the goal coverage of the existing metrics

The existing metrics cover a total of nine different goals, which the catalog's goal-metric mapping generates. Table 6.1 lists the result of the goal coverage analysis. According to the second column of the table, two-thirds of the goals fall into the development organization and processes category. Only two of the goals relate to the product and one to finance. The existing metrics do not cover goals from the customer and employee categories. The third column of the table shows the percentage share of every goal in the existing metrics. It shows that the goals of transparency and predictability are already very well covered. More than half of the metrics also cover the improved planning goal. Twenty-nine percent of the metrics address the goals of delivery reliability and productivity. The goals of quality, cost management, speed, and agility are covered less strongly by only one metric each.

Step 3: Collect and prioritize agile software development goals

The same interview partners of the case study also served as representatives of the IT-provider in this step. The roles of the representatives can be found in Table 4.1. In total, the representatives named ten different goals and elaborated on their purpose, object, and perspective. By far, the most frequently mentioned goal is quality, which came up in all interviews in varying degrees of priority. Quality is what the customer values. One aspect of their valuation is the freedom from defects that can be realized via sufficient testing. The stability of the application also belongs to the absence of defects. Another aspect is

Goal	Goal category	Goal coverage in %
Transparency	Development organization & processes	71 %
Predictability	Development organization & processes	71 %
Improved planning	Development organization & processes	57 %
Productivity	Development organization & processes	29 %
Delivery reliability	Product	29 %
Cost management	Finance	14 %
Quality	Product	14 %
Speed	Development organization & processes	14 %
Agility	Development organization & processes	14 %

Table 6.1.: Percentage goal coverage of existing metrics at the IT-provider

the realization of the desired requirements. The service, meaning the speed of reaction to defects, is also part of the quality. All three aspects contribute to a higher acceptance of the manufactured software product and higher customer satisfaction. The goal of the quality corresponds to the following GQM-template sentence:

> "Improve (=purpose) the quality (=issue) to increase satisfaction with delivered features (=object) from the customer's point of view (=perspective) in the context of the IT-provider (=context)."

The goal of quality is also indirectly related to the goal of commitment, for example, when it comes to meeting delivery deadlines. Because, from the customer's point of view, meeting delivery deadlines can also be an aspect of quality. The interviewees mentioned this goal three times in one form or another. Commitment refers to the three dimensions of time, quality, and cost. The IT-provider needs to be reliable in all three dimensions. Information-based planning and forecasting contribute to improved commitment. Commitment starts with estimation and runs through all phases of agile software development up to delivery. It also includes post-delivery maintenance. Planning and forecasting also influence communication with the customer. Binding adherence to deadlines and delivery of the agreed service improves the mutual trust between the two parties. In addition, it is important to make communication with the customer transparent and to communicate changes in planning well in advance. Besides transparency, the interviewees also mentioned the goal of fast delivery twice, which overlaps with the time component of commitment. Like quality, fast and on-time delivery contributes to customer satisfaction. All these considerations of the interview partners find their way into the following formulation:

> "Improve (=purpose) the commitment (=issue) with regard to the on-time, on-cost, and performance-agreed delivery of software by the agile software teams (=object) from the customer's point of view (=perspective) in the context of the IT-provider (=context)."

The third goal within the GQM-plan is security. The interview partners mentioned this goal twice. However, since it was regarded as having lower importance in each case, it has a lower priority than the goals of quality and commitment. The goal is particularly relevant due to the regulatory requirements in the insurance industry. Therefore, the organization wants to increase the level of security constantly. Stability and security are also in the customers' interest, who expect secure software that complies with regulatory requirements. According to the GQM-template, the following formulation can be derived:

> "Increase (=purpose) the security (=issue) in software products in terms of stability and meeting regulatory requirements (=object) from the customer's point of view (=perspective) in the context of the IT-provider (=context)."

In addition to the three goals that are part of the GQM-plan based on the frequent mentions, the interviewees also mentioned other goals. However, these goals only occurred once each, so they were omitted to preserve focus. On the one hand, this concerns customer satisfaction, which the IT-provider does not measure directly so far. In the long term, the organization would like to establish a metric in the form of a customer survey to track the goal more concretely. The details are still pending. Second, management would like to see metrics that measure profitability. The implementation of measuring the goal of increasing profitability is not fixed and needs to be clarified in the longer term. Furthermore, the interviewees mentioned employee satisfaction, transparency in requirements quality, regular delivery, and predictability of capacity planning as goals. These goals are not directly part of the GQM-plan and were deliberately omitted.

The three final goals are shown in Table 6.2, together with the developed questions and selected metrics of the GQM-plan.

Step 4: Develop questions with hypotheses

In connection with three goals from step 3, I talked with interviewees about their information interests. The interview partners were asked to formulate their information interests concerning the goals in concrete questions. I collected the questions and tried to find overlaps. This process resulted in seven different questions that operationalize the goals.

The information interest related to quality breaks down into four different dimensions. The first dimension directs the question to defects per unit of work. Here, the interviewees explicitly mentioned the defects after the release that the customer detected. Nevertheless, also the defects during development and for the various customer groups are relevant. Therefore, the first question asks very generally about the quality of the features, whether before or after deployment.

> "How qualitative are the features?"

The second question is about the classification of defects in terms of criticality. The acting persons have the interest to know how many defects reach a critical level.

"How critical are the defects?"

This information can be used in the next step to ask about the speed for handling critical but also defects in general. The speed in processing defects is considered as a partial aspect of quality.

"How quickly are teams able to respond to (critical) defects?"

The last point of information interest is directed to the degree of automation of tests. The term tests refers to regression tests that the teams perform in different test management tools.

"How automated are the tests?"

The goal of commitment is to ensure that the acting stakeholders, such as the Product Owner and the Service Owner, have reliable data about development and the release. They can then use this information in the second step to inform management and, above all, customers about progress and delivery. Commitment includes, among other things, keeping agreed target dates, planning precise work times and assessing their influence factors, and succeeding in delivering the scope of a release.

"How committed is the planning and its communication to the customer?"

The second part of the information interest of the commitment goal lies in understanding the processing time of software features. With this information, stakeholders can better estimate what scope and delivery date to target. The period for the processing time extends from the order to the delivery. This information interest reflects the speed goal as well.

"How fast is the development?"

Even though the information interest is high concerning security, the possibilities to measure the goal by metrics in agile software development are limited. Nevertheless, there is a need to find out how large the share of the development effort is that the agile teams invest into the security of the software products.

"What proportion of the capacity spent is devoted to security?"

The questions mentioned in this step are all part of the GQM-plan and can therefore be seen in Table 6.2. The metrics team has not yet come up with hypotheses for the questions because the feasibility of the metrics was a higher priority for the time being.

Step 5: Search metrics candidates and select metrics

The analysis of the current use of metrics revealed that the agile teams primarily use metrics at the team level. With the increasing number of Scrum teams and the subsequent scaling of agile practices, it is obvious to make greater use of the possibilities of metrics at the cross-team level as well. On the one hand, the metrics team expects increasing transparency about further goals, and on the other hand, it wants to improve the basis for decisions and communication. In addition, the metrics team envisions the benefits of introducing new cross-team metrics in more accurate planning and forecasting concerning releases, which will enable better management and coordination of the agile teams. For these reasons, the metrics team, led by the Scrum Master, decided to introduce and actively use a total of 13 new metrics. Eleven of them are cross-team metrics. Compiling the metrics candidate list, the metrics catalog, Scrum@Scale Guide, and interviews served as input sources. Since there was initially no goal prioritization from the IT-provider, the metrics candidate list was extensive and included over 120 metrics. After prioritizing the goals and limiting them to the three goals of the GQM-plan, the list contained 75 metrics. In the end, the metrics team leader selected eight of these metrics. Six of these eight metrics were suggested simultaneously by the metrics team members during the interviews. Five metrics came exclusively from the interviews or were individually adapted to the needs of the agile teams based on metrics from the catalog. The high proportion of metrics from the interviews demonstrates that this approach ensures that the wishes of the team members are given strong consideration.

For the first goal, the keyword of the same name was chosen at level 2 of the metrics catalog to find quality metrics. The resulting candidate list contains 42 metrics, of which the metrics team selected four. These include the *defects per feature after deployment* metric, which the metrics team collects at a cross-team level. The metric measures the quality of features by counting the number of defects per feature after deployment. In the context of the IT-provider, a feature is a requirement that can consist of several epics and, thus, several user stories. Defects are generally a collective term in the context of the IT-provider for either error in development, which is also called test deviation, or error in production, which is called a bug. In the case of this metric, the latter is applicable. The second metric is from the catalog but was also mentioned in the interview, can be found in the Scrum@Scale Guide, and refers to the team-level. The metric measures *defects per user story*. The sprint is the period to which the metric relates. Defects mean the test deviations per user story during the development phase. The third metric is similar, although it does not concern a user story but rather the larger unit of epics. *Defects per epic* measure the quality of all teams working together on the epic. Again, the defects are quantified here as test deviations. The IT-provider has different customers offering customized solutions under multiple brands. Therefore, the IT-provider allocates its software packages to customer environments. To measure the quality of the customer environments, the metrics team wants to count the defects from development and production attributable to them. The self-constructed metric provided for this purpose is called *defects per customer environment* and measures on a team-wide basis. Furthermore, the metrics team wants to output the *amount of defects per criticality category*. There are four gradations from highest to high to medium to low. The metric does not recognize team boundaries and includes defects like test deviations and bugs. The sources of the metric are the metrics catalog and the interviews. The following two metrics intend to answer the question about the speed of the agile teams for fixing the defects. For this purpose, the metrics team uses the criticality categories in each case. One metric measures the *average defect lead time per criticality* *category*, and the other measures *the average defect cycle time per criticality category*. Although both metrics relate to the cross-team level, the former measures from ticket creation and the latter measures from the start of processing until the defect is resolved. The calculations are made by summing the average times of all defects per category divided by the number of defects per category. Both metrics build on existing metrics in the catalog but extend them with the criticality category. The last quality metric is related to the automation level of the regression tests. The *percentage of automated tests* is calculated by dividing the number of automated tests by the total number of tests performed. A test is considered automated if, after being set up once, the test team can repeat the test an infinite number of times without any additional effort. The regression tests refer to several teams. The metric can be found in the catalog.

The goal commitment cannot be found directly in the catalog. Nevertheless, I compiled a candidate list of 36 metrics via the level 2 search term proxies predictability, delivery reliability, and efficiency. The metrics team selected four from this list. The first two metrics intend to answer the question of commitment in planning and communication. The *epic burn-down chart* measures the remaining work of an epic on a team-wide level in relation to time. The metric visualization is a chart with a horizontal axis indicating the time and a vertical axis indicating the work remaining. As epics are completed and set to done, the line indicating the remaining work decreases. An epic consists of several user stories and currently has no time fixation, as is the case with the user story per sprint. The *release burn-down chart* visualizes the remaining work for a release in relation to the remaining time. The visualization of the metric is a chart with a horizontal axis showing sprints and a vertical axis showing remaining work in story points. Analogous to the epic burn-down chart, the line decreases when user stories are completed. Since a release can apply to multiple teams, the metric is also team-wide. The metrics team hopes to provide more reliable statements to customers with the two metrics *feature cycle time* and *average user story cycle time*. The metric is a measurement of the time it takes to work on a feature from the start until it is set to done. Concerning a user story, the average cycle time divides the sum of all cycle times by the total number of user stories. This metric refers to the team level.

The only metric referring to the goal of security is *security-related story points per sprint*. The metric comes from the interview with the Information Security Officer who expressed this need. The metric requires the assignment of security labels when a user story contributes to security in the software products. The sum of all story points of these labeled user stories will result in the metric value per sprint. The sum is calculated across all teams. The exact definition for the security label still needs to be clarified. Apart from that, we have not collected any further suggestions for the goal of security.

All metrics mentioned above are shown in Table 6.2 as part of the GQM-plan.

Step 6: Prepare the data collection

The responsibility for preparing the data collection lies primarily with the metrics team leader. According to the GQM-plan, this person tries to set up the metrics in Jira. The data collection can start as soon as the setup and configuration have been completed. The metrics team

Goal	**Question**	**Metric**
Improve the quality to increase satisfaction with delivered features from the customer's point of view in the context of the IT-provider	How qualitative are the features?	Defects per feature after deployment
		Defects per user story
		Defects per epic
		Defects per customer environment
	How critical are the defects?	Amount of defects per criticality category
	How quickly are teams able to respond to (critical) defects?	Average defect lead time per criticality category
		Average defect cycle time per criticality category
	How automated are the tests?	Percentage of automated tests
Improve the commitment with regard to the on-time, on-cost, and performance-agreed delivery of software by the agile software teams from the customer's point of view in the context of the IT-provider	How committed is the planning and its communication to the customer?	Epic burn-down chart
		Release burn-down chart
	How fast is the development?	Feature cycle time
		Average user story cycle time
Increase the security in software products in terms of stability and meeting regulatory requirements from the customer's point of view in the context of the IT-provider	What proportion of the capacity spent is devoted to security?	Security related story points per sprint

Table 6.2.: Goal, Question, Metric Plan of the IT-provider

leader also assigns the permissions for the dashboard where the metrics will be visible. This way, all stakeholders can access the new metrics. The display of the new metrics requires that the base metrics are set up in Jira. The base metrics are the foundation for calculating and visualizing the data collection of the metrics. Table 6.3 shows the required base metrics and the associated components of the defined measurement plan. These include the object, scope, time or event of data collection, source, responsible, and means for each metric. The first four base metrics in the table are identical to those in Table 6.2. The objects of the four are a feature, user story, epic, and customer environment. As with all of the following base metrics, the scope refers to the IT-provider's agile Scrum teams. Jira collects the data for the defects per feature after deployment metric automatically for each deployment. For the other three base metrics, Jira gathers the data continuously and in an automated manner. The lead time per defect, cycle time per defect, and criticality category per defect serve as the base metrics for the similarly named average metrics per criticality category. The former two collect their data after each defect is completed. The criticality category is always set by the developers when the ticket gets created. The object is the defect for all three metrics, and Jira automatically collects the data for all of them as well. The two base metrics, number of total tests and number of automated tests, are necessary for calculating the aggregated metric percentage of automated tests. On the one hand, the metrics refer to all regression tests and, on the other hand, to all automated regression tests. Various tools such as Cucumber, Cypress, or the Jira plug-in Xray collect the data automated for each release. Cucumber is a tool that is hard to maintain and will be replaced by Cypress over time. Xray provides the ability to integrate the data into the dashboard of the other metrics in Jira. The responsibility for data collection of the test base metrics is taken by a test manager prospectively. Behind the epic and release burn-down chart are the base metrics of not completed epics and story points per time. Jira measures these automatically and continuously as predefined metrics. Therefore, there is no need for configuration in these two cases. The feature cycle time and average user story cycle time can also be transferred to the dashboard without any significant effort from the metrics team leader. Jira performs the data collection automatically and continuously. The last metric security-related story points per sprint is the responsibility of the Information Security Officer. Defining the requirements for the security label is also in these hands. Per sprint, the metric is supposed to be calculated automatically in Jira. The overall responsibility for the dashboard and the metrics it contains belongs to the metrics team leader. In case of adjustments to the dashboard, the metrics team leader is responsible and acts as a contact person for suggestions for improvement. The metrics team relies on existing data for data collection that is automatically generated in Jira triggered by the actions of the agile teams.

Step 7: Collect, analyze and interpret the metrics data

The introduction of the metrics and the corresponding data collection will take place iteratively over two months. Week by week, the metrics team leader introduces more metrics. Jira automatically collects most metrics continuously throughout a sprint. The data collection requires no manual effort after the one-time configuration by the metrics team leader. The configuration includes integrating the metrics into a central dashboard that gives team

Base metric	Object	Scope	Time/event	Source	Responsible	Means
Defects per feature after deployment	Feature	Agile teams	Per deployment	Jira	Scrum Master	Manuel
Defects per user story	User story	Agile teams	Continuously	Jira	Scrum Master	Automatic
Defects per epic	Epic	Agile teams	Continuously	Jira	Scrum Master	Automatic
Defects per customer environment	Customer environment	Agile teams	Continuously	Jira	Scrum Master	Automatic
Lead time per defect	Defect	Agile teams	After set to done	Jira	Scrum Master	Automatic
Cycle time per defect	Defect	Agile teams	After set to done	Jira	Scrum Master	Automatic
Criticality category per defect	Defect	Agile teams	After set to done	Jira	Scrum Master	Automatic
Number of total tests	Regression test	Agile teams	Per release	Xray	Test manager	Automatic
Number of automated tests	Automated regression test	Agile teams	Per release	Xray	Test manager	Automatic
Not completed epics per time	Epic	Agile teams	Continuously	Jira	Scrum Master	Automatic
Not completed story points per time	Release progress	Agile teams	Continuously	Jira	Scrum Master	Automatic
Feature cycle time	Feature	Agile teams	Continuously	Jira	Scrum Master	Automatic
Average user story cycle time	User story	Agile teams	Continuously	Jira	Scrum Master	Automatic
Security related story points per sprint	Story point	Agile teams	Continuously	Jira	Information Security Officer	Automatic

Table 6.3.: Measurement plan of the IT-provider

members an easily accessible and transparent view of the Scrum-at-scale team's development. Visualization is also part of the configuration. For this purpose, the metrics team leader chooses from Jira's vast pool of gadget possibilities. For example, the defects per customer environment metric was implemented via a pie chart gadget and added to the dashboard. The required aggregation of data for some metrics, such as average defect cycle time per criticality category, also takes place within the configuration in Jira. Checking the data for completeness, internal consistency, external consistency, plausibility, and the separate handling of abnormal data is the overall responsibility of the metrics team. The analysis and interpretation of the metrics values are carried out as part of the sprint review event. The Scrum Master evaluates the past sprint with the Product Owner and the developers of all Scrum teams. The metrics dashboard acts as a platform for analyzing and interpreting the results. Another event where the analysis and interpretation occur is the security board for the metrics assigned to the security goal. The Information Security Officer chairs the security board.

Upon completion of the introduction, the metrics team leader presents the results to management. The presentation should include an analysis of the goal achievement and answers to the questions posed by the values of the metrics. The analysis will determine the continuance of each metric and possibly additional action required in the form of new metrics or modified goals.

Step 8: Check the achievement and coverage of goals regularly

After the one-time management review of the adoption of the metrics, the metrics team will regularly perform step 8 in the sprint retrospective meeting every two weeks as needed. The sprint retrospective meeting is similar in composition to the sprint review meeting. The Scrum Master and Product Owner participate as well as all members of the agile Scrum teams. In this format, the participants can discuss the goals and their coverage by the metrics and determine what could be changed to measure the goals even more precisely and find more suitable metrics for the next sprint. As the implementation of the Scrum-at-Scale framework progresses and the number of Scrum teams increases, the scaled version of the sprint retrospective meeting can also serve as an agile event for this step in the future.

6.1.2. Improvement of the measurement program management

Step 1: Assess the metrics team maturity level

The maturity of the metrics team was sent out as a survey to all IT-provider interview participants with a request to complete the template. Participants received either a descriptive explanation during the interview or by mail on how to complete the template. The participants then filled out the template remotely on their own and sent it back to us. In the end, we received the completed template back from three participants. Among them were the Scrum Master, Product Owner, and Chief Technology Officer.

Step 2: Evaluate the metrics team maturity level

Even though the teams measure agile metrics from the outset, the measurement program has maturity potential, especially at the cross-team level. Since the number of agile teams is in the process of being established, many of the statements assessed were not yet necessary and are only about to be implemented. The metrics team faces the challenge of accepting its organizational mandate and expanding the measurement program to the program level. This plan requires professionalization actions. Figure 6.1 shows the current maturity of the metrics team. The extent of the blue line in the diagram indicates the maturity level in the six dimensions. Overall, participants answered six percent of the statement with fully compliant. This assessment was relatively rare because it required at least two participants to check the respective box. The answer option partially compliant, by contrast, was ticked by the participants in 55 percent of the cases because, in addition to the majority of answers, this category was also used for very diverging answers. For example, if one participant was fully compliant with the statement, a second was partially compliant, and the third rated it as missing, we assessed the response also as partially compliant. The statements that participants considered to be missing amounted to 33 percent. This number highlights the potential for improvement in terms of maturity. Five statements are not relevant for the metrics team, resulting in a share of seven percent. Again, at least two participants had to tick this answer option. In other words, the statement is relevant if two or more participants put their cross in the three answer options: fully compliant, partially compliant, or missing. The radar chart reveals that team performance and efficiency are solidly developed. The area values of set-up, commitment, impact, and longevity, however, range between 17 and 27 percent. These areas need improvement. The set-up area has a maturity level of 28 percent, which is still the best among those requiring further actions. Whereby the assessment for the sub-areas is very heterogeneous. Formalization and roles in the organization have values of 42 and 38 percent, respectively, whereas the area roles in the team only reaches eight percent. In the second area, the picture is similarly dissimilar between the sub-areas. The first sub-area concerns the commitment of the team, which is rated at 30 percent. The organization's commitment, in contrast, is almost non-existent at zero percent. Statement 22, which deals with management support for the metrics team, is not relevant. The third area, which assesses the impact of the metrics team on the organization and the outside world, scores 25 percent. The respective sub-areas show values of 33 percent and ten percent. The area longevity has the worst score of all areas with 17 percent. In addition, a third of the nine statements are not relevant for the metrics team. Among them are statements 41, 45, and 47, which deal with surviving organizational change, the metrics team as an interesting place to work, and career opportunities. The fifth area evaluates the efficiency and ways of working of the metrics team. It complies with fifty percent of the relevant statements. The short lead time for establishing new measures and information products in statement 50 is not relevant. The last of the six areas has the highest maturity level at 58 percent. Here, the model assesses team performance. In the team performance sub-areas context, mission, talent, norms, and buy-in, the metrics team complies with 50 percent of the statements. The maturity level for the resources and courage sub-areas is 67 and 75 percent, respectively.

The exact results per statement can be found in Tables A.7, A.8, A.9, A.10, A.11, and A.12 of Appendix A.3. The final results of the maturity assessment were shared with all five interview participants.

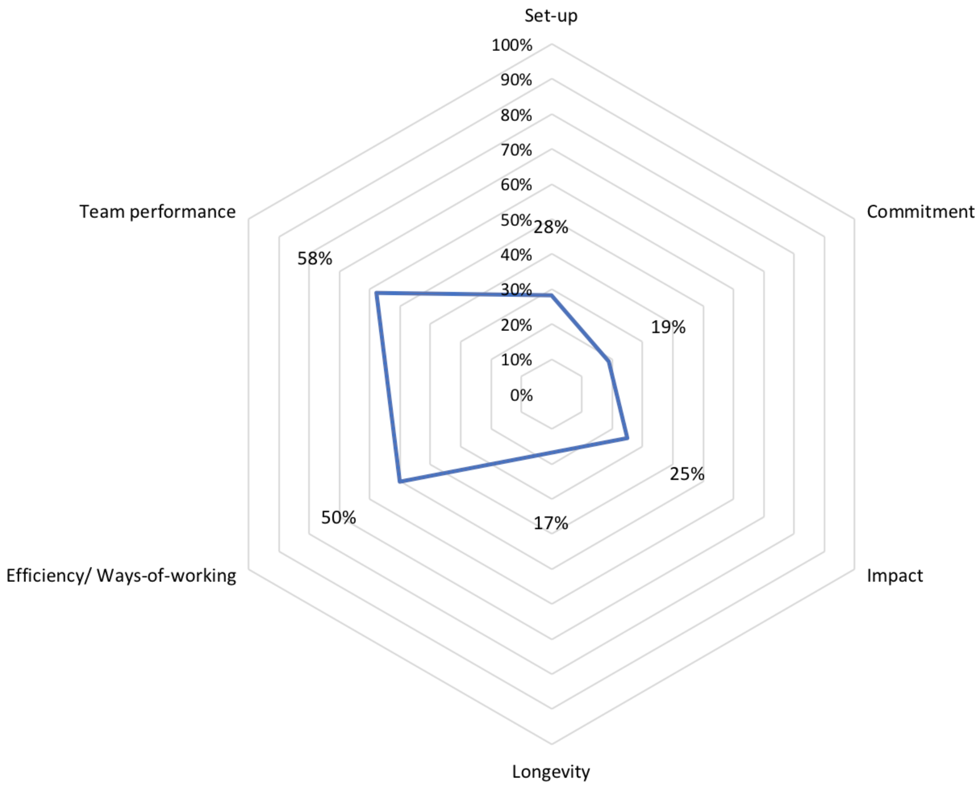

Figure 6.1.: Radar diagram of the IT-provider metrics team maturity assessment

Step 3: Plan and implement actions to improve the maturity level

The IT-provider is currently transforming many areas and is carrying out a series of optimizations. Therefore, the focus is currently on other initiatives and does not allow for active planning and implementation of steps within the scope of the assessment for the time being. Nevertheless, according to the members, the evaluation of the maturity level of the metrics team has helped to gain a deeper understanding of the current state of the metrics program. The metrics team member will use the assessment as a basis for discussing actions to improve the management of the metrics program when the appropriate opportunity arises. According to the members, there is direct potential for improvement in the areas of commitment and set-up in particular.

Step 4: Perform the metrics team maturity assessment regularly

According to their statement, conducting the assessment followed by planning and implementing actions in three monthly cycles would be ideal for the IT-provider. However, many initiatives have a higher priority, making a regular execution of step 4 at intervals of one year seem more realistic. The group that regularly executes the approach and decides on the actions is the same as the interview participants.

6.2. Execution of the approach at the Agile Release Train

Involved stakeholders and their role

Metrics and related decisions are a substantial part of a weekly steering meeting between the RTE, Product Manager, System Architect, and the Sales ART's primary manager. The latter is considered a neutral authority that acts as an organizational leadership tie-breaker in case of stalemate and is involved in decision-making on important issues. The Release Train Engineer may be considered an unofficial metrics team leader, although all decisions are made in direct consultation with the other members of the committee. The RTE also acts as a connector between the steering committee and the Scrum of Scrums meeting. The Scrum of Scrums meeting includes all Scrum Masters of the ART. This meeting provides the opportunity to discuss metrics introduction projects and to advocate for acceptance. In addition, the members review the outcomes of the metrics and discuss feedback from the developers. Hence, the metrics team virtually spans these two groups and includes both the members of the steering committee and the Scrum Masters of all agile teams.

6.2.1. Goal-based introduction of metrics

Step 1: Identify existing metrics

The first step was to identify existing metrics via interviews with individuals from a wide variety of roles. The interviewees came from both the steering committee and the agile teams. The existing metrics are calculated based on different currency bases. Firstly, the organization uses surveys to generate statistical analyses of customer and employee satisfaction. Furthermore, the ART relies on booking person-days to calculate the metrics effort allocation, commitment quota, and run & change per PI and sprint. The ART also supports developers with several test metrics such as error tickets, error ticket fixing and response time, and test coverage. These are calculated as a sum, duration, or in the form of proportional lines of code. The metrics throughput and sprint burn-down chart rely on stories, epics, or story points. The domain of metrics with this currency basis is expandable in the future. Table 4.5 in Section 4.4 provides an overview of the existing metrics of the ART and a description of the calculation rule. The table also shows that most metrics are used at the program level and that the team level is less well covered. Moreover, there is a lack of horizontal alignment at the team level, so there is no common standard for metrics at this level. This lack of horizontal alignment is due to the different agile maturity levels of the teams and the differing progress

Goal	Goal category	Goal coverage in %
Customer satisfaction	Customer	46 %
Transparency	Development organization & processes	38 %
Quality	Product	23 %
Customer communication	Customer	23 %
Cost management	Finance	23 %
Improved planning	Development organization & processes	15 %
Customer centricity	Customer	15 %
Productivity	Development organization & processes	15 %
Contract fulfillment	Product	15 %
Delivery reliability	Product	8 %
Employee satisfaction	Employee	8 %
Employee motivation	Employee	8 %
Adherence to deadlines	Development organization & processes	8 %
Agility	Development organization & processes	8 %

Table 6.4.: Percentage goal coverage of existing metrics in the ART

in internalizing agile working. In the future, therefore, the same metrics should be used across all agile teams in the ART to enable a collective basis for comparison. Prerequisites for this are the estimation in story points and the uniform use of tools such as Jira.

Step 2: Determine the goal coverage of the existing metrics

Based on the metrics mentioned in step 1, we determined the goal coverage using the metrics catalog. Table 6.4 presents the assignment to the goal category from the catalog and the respective coverage in percent for each goal [111]. It can be observed that the customer satisfaction goal directly or indirectly relates to almost half of the metrics. The progress transparency goal also achieves comparatively high coverage of 38 percent. Nearly every fourth metric covers quality, customer communication, and/or cost management. Fifteen percent of the metrics address the goals of improved planning, customer-centricity, productivity, and contract fulfillment. The goals of delivery reliability, employee satisfaction, employee motivation, adherence to deadlines, and agility have the lowest coverage of one metric each. Most of the goals pertain to the category development organization and its processes.

Step 3: Collect and prioritize agile software development goals

Five representatives were interviewed for the collection of the ART's agile software development goals. The representatives fill the Release Train Engineer, Scrum Master, Developer, and Product Manager roles, as shown in Table 4.1 in the methods section of the case study. These respondents identified a total of eleven different goals during the interviews. The highest priority goal is quality mentioned by three interviewees and prioritized highest by one of the interviewees. Hence, quality belongs to the GQM-plan. The purpose is to improve. The issue is quality in terms of software code and conceptual design as an object. The goal reflects the perspectives of the standards-setting management and the standards-compliant agile development teams in the Sales ART context. Additionally, the quality goal follows the idea of sustainable development and avoidance of maintenance effort. According to the GQM-template, the complete goal is as follows:

> "Improve (=purpose) the quality (=issue) with respect to software code as well as conceptual design (=object) from the perspective of management and agile development teams (=perspective) within the ART (=context)."

Furthermore, the respondents added collaboration, commitment, communication, transparency, and customer satisfaction to the goals collection. There are two mentions of each of these goals. Even though one interviewee prioritized customer satisfaction the highest, the goal is very well covered by 46 percent of the existing metrics, as seen in the previous step. A survey even measures customer satisfaction directly along various dimensions. Therefore, there is no need for additional metrics that explicitly measure customer satisfaction. Communication and collaboration are two strongly interrelated goals. Both are difficult to quantify and do not appear directly as goals in the catalog. Nevertheless, the metrics team is concerned with pursuing the goals and obtaining a better basis for decision-making via metrics. The purpose is to improve the issue of lack of direct and fact-based collaboration. Communication should be stimulated so that dependencies within the team and across team boundaries are recognized and overcome. According to the interviewees, establishing an evidence-based feedback culture within the ART will play a crucial role in the future. The formulated goal with the building blocks of the GQM-template is:

> "Improve (=purpose) the collaboration (=issue) to accelerate work processes and stimulate direct communication between teams and team members (=object/perspective) within the ART (=context)."

Another goal with two interview mentions is commitment. As a third goal on the GQM-plan, the purpose is to increase. The issue concerns the binding delivery of work packages in the planned time with the planned scope and content. This follows the idea to create the necessary trust among the customers but also the members of the ART. A factor for this is sound, realistic, and information-driven planning. In summary, the goal is:

> "Increase (=purpose) the commitment (=issue) to build trust and better plan work packages in terms of their content and scope (=object) from the perspective of customers and members (=perspective) within the ART (=context)."

As a fourth goal, the metrics team wants to improve transparency about their workflows. The team hopes this will benefit collaboration when metrics enable more transparent communication. Therefore, the goal primarily addresses the members of the ART. The steering committee expects better transparency about the individual phases of agile software development and progress in implementing the requirements. The entire goal states:

> "Improve (=purpose) the transparency (=issue) of the status of workflows and requirements (=object) from the perspective of all members (=perspective) within the ART (=context)."

Since a fourth goal exceeds the recommended number of three and 38 percent of the existing metrics cover the goal already, the team limited the search for new metrics for this goal. The other goals did the interviewees only mention once and consequently not transferred to the GQM-plan. These include the goals of responsibility, skill-building, breaking up monoliths, reusability, and flexibility. The team decided to prioritize the four previously described goals higher and, thus, limit the metrics candidate list in the following steps. Nevertheless, the goal of reusability partially reflects through the quality goal. Finally, not all goals that the interviewees mentioned map quantitatively. An example of this is team building and internal personnel development that aims to promote the reduction of head monopolies. These goals can be mapped less by quantitative metrics than by qualitative measures.

The four final goals, however, are part of the GQM-plan. The first columns of the two divided GQM-plan tables 6.5 and 6.6 show the four goals with the questions and metrics presented in the following steps.

Step 4: Develop questions with hypotheses

Together with the interview participants, I discussed matching questions in addition to the goals and metrics used so far. I collected the questions across all interviews and determined the intersections in the information interests behind the goals. For one, knowledge of what the customer expects qualitatively from the product is essential for the respondents. Product quality refers to the code quality and the offered functionality. Respondents also want to ensure that the teams meet the requirements without defects. To this end, stakeholders need to know how many tests the teams perform, how many lines of code the tests cover, and what the relationship is between effort expended and increase in quality. Minimizing the effort can be achieved, for example, by a high degree of automation in the tests. Another information need relates to the reusability of parts of the software. This collected information interest leads to four final questions, which link the quality goal and the associated metrics.

> "How many defects occur per sprint?"
>
> "How extensive is our test coverage?"
>
> "How automated are the tests?"
>
> "How much software code is reused?"

The information interest regarding the goal of improving collaboration addresses the frequency of direct communication. This concerns the communication of the members of the own team but also the communication across team boundaries. Direct communication also includes regular feedback. Respondents indicated an interest in how satisfied teams are with collaboration and communication. Information about the ability to solve problems with each other and to help each other is also vital. The Scrum Masters have a justified interest in monitoring the mood of the team. The following question tries to capture all these aspects:

> "What are the team's relative strengths and weaknesses in collaboration and communication?"

Behind the commitment goal is the need for information about the adherence to sprint planning and PI planning in terms of content, scope, and achievement. The interviews showed that it is a priority for the agile teams to know whether the committed goals and tickets have been completed. The information interest also targets the disruptions that can negatively impact commitment. The commitment goal breaks down at the operational level specifically into four questions.

> "How committed is the sprint planning of the teams in terms of content and scope?"

> "How committed is the team to achieving the sprint planning?"

> "How committed is the ART in achieving the PI planning?"

> "How do disruptions affect commitment?"

The informational interest of the transparency goal is specific to workflows and asks about the amount of work in the states of the development process. Stakeholders want to know how many requirements have been tested but not released. The progress of the release is of particular interest to the respondents. The information interest for the transparency goal summarizes the following question:

> "How well do the workflows function within a PI?"

The whole ten questions are part of the GQM-plan and can be found in tables 6.5 and 6.6. The metrics team did not use hypotheses for the time being to avoid overwhelming the agile teams with a large number of requirements and changes.

Step 5: Search metrics candidates and select metrics

The existing metrics in the ART have a strong focus on the program level and person-days as the basis for calculation. Team-level agile metrics with story points as a basis for calculation are inconsistently used across teams and not widespread overall. The metrics team wants to change that and introduce a consistent baseline of agile metrics. The intention is to manage the metrics on a scaled level in a dashboard and aggregate program metrics from the team metrics in the long term. To realize this intention, the metrics team introduced 19 metrics in the short, medium, or long-term as part of the goal-based approach. The SAFe Reference Guide and metrics catalog have been instrumental in compiling a candidate list of metrics.

Based on the identified goals and the questions reflecting the information interest, 54 agile metrics were proposed to the metrics team. For the quality improvement goal, 28 metrics were picked out. Twenty-five of the metrics came from the metrics catalog and were filtered using search term quality. Seven metrics came from the SAFe Reference Guide. These seven quality metrics, four of which are also part of the catalog, were all ultimately selected by the metrics team. The first metric measures the *number of defects per sprint*. The metrics team plans to introduce this metric in the medium term. The second metric is part of the long-term introduction plan and provides information about *unit test coverage*. To do this, the metric puts the number of lines of code that all unit tests are currently running in relation to the total number of lines of code in the software under test. Two other selected metrics likewise address test coverage. One metric measures the *number of new test cases*, and the other measures *total tests*. Both metrics are short-term. The next two metrics address the automation of tests. The *percentage of automated tests* compares automated tests with the total number of tests on a software product. The second automation metric counts the *number of new automated test cases* during a sprint per team. The last of the seven selected quality metrics measures the *number of refactors*, which is the number of modifications to a software unit - a module, method, or program - undertaken by a team during a sprint. The metric not only corresponds to the goal of quality but also pursues the goal of reusability. The metrics team classified the number of refactors as a short-term metric. All of these seven quality metrics are agile team metrics and refer to the time unit of a sprint. Since the second goal of improving collaboration is not found one-to-one in the metrics catalog, I tried to find metrics candidates using the search terms employee satisfaction and motivation. However, no suitable metric candidates could be found this way. The SAFe Reference Guide, in contrast, includes the *Team Self-Assessment*, which allows the agile teams to improve team health and processes. Team members anonymously rate product ownership, PI health, iteration health, team health, and technical health on a scale of zero to five. Each of the five dimensions has different questions that each team member answers based on the scale. The assessment results in a radar chart that shows the team's relative strengths and weaknesses. The team health dimension is of particular interest for achieving the goal of improving collaboration. The metrics team plans to introduce the assessment in the medium term. The eight metrics mentioned so far cover two of the four goals. Table 6.5 shows these two goals and eight metrics along with the intermediate questions according to the GQM-plan.

The second part of the metrics concerns the goals of increasing commitment and improving transparency. The former does not reflect in level two of the metrics catalog. However, the goals of adherence to deadlines, contract fulfillment, and delivery reliability were used as proxies as filters. The filtering resulted in 23 candidate metrics. The metrics team chose six of these metric candidates. Furthermore, the team selected four other metrics from the SAFe Reference Guide. The first two metrics measure the agile teams' commitment to sprint planning in content and scope. Both fall into the short-term category. The first metric counts the *number of added stories*, meaning those that the agile teams subsequently include during a started sprint. Ideally, the value should be zero. The second metric measures the *number of stories planned* by a team, in other words, committed before the sprint. In addition, the selected

Goal	Question	Metric	Term
Improve the quality with respect to software code as well as conceptual design from the perspective of management and agile development teams within the ART	How many defects occur per sprint?	Number of defects per sprint	Medium
	How extensive is our test coverage?	Unit test coverage	Long
		Number of new test cases	Short
		Total tests	Short
	How automated are the tests?	Percentage of automated tests	Short
		Number of new automated test cases	Short
	How much software code is reused?	Number of refactors	Long
Improve the collaboration to accelerate work processes and stimulate direct communication between teams and team members within the ART	What are the team's relative strengths and weaknesses in collaboration and communication?	Team Self-Assessment	Medium

Table 6.5.: Goal, Question, Metric Plan of the ART with introduction term: Part I

metrics list contains four metrics that measure the achievement of sprint planning. These include the medium-term metric *sprint burn-down chart*. Some agile teams already use the metric, but the uniform implementation as a standard is missing. The metric intends to help the Product Owner understand the team's progress in a sprint and thus additionally promote collaboration on both sides. The metrics *number of not accepted stories* (remaining work as the catalog calls it) and the *number of accepted stories* count the stories that are (not) defined, created, tested, and accepted by a team within a sprint. From both metrics, the *percentage of accepted stories* can be calculated. This metric is considered to measure the achievement of a team's planned effort. All three metrics focus on the short term. At the program level, the metrics team would like to use this data set to introduce the *PI burn-down chart* (also called the Epic burn-down chart in the catalog) in the medium term. The last three metrics of the commitment goal get to the root cause of commitment disruptions. These three metrics are classified as short-term and relate to the team level. One of the metrics measures the *number of not accepted stories in a sprint that the team pushed to the next iteration*. Another one counts the *number of stories not accepted and deferred to a later date*. The last one records the *number of stories not accepted and deleted from the backlog*. Moreover, with the knowledge of the cause of the disruptions, the agile teams can better collaborate based on information in a targeted manner. The metric for the fourth goal of improving transparency was not found through a targeted search using the metrics catalog but stemmed from the metrics team's desire to introduce this metric over the long term. The lean program metric appears in the metrics catalog and the SAFe Reference Guide and is called the *cumulative flow diagram*. It measures the ART's workflow during a sprint. The visualization of the metric is a chart with a horizontal axis indicating the time and a vertical axis indicating total work. Time follows the sequence of agile software development phases such as backlog, implementation, deployment to production, release, and done. The metrics team envisions this story- or feature-based metric as complementary to the existing program-level transparency-enhancing metrics, but which are primarily based on person-days. The second part of the GQM-plan presents Table 6.6, including the metrics just mentioned and their terms.

Step 6: Prepare the data collection

Together with the Scrum Masters and the Release Train Engineer, we planned the preparation of the data collection in a regular Scrum of Scrums meeting. As the unofficial metrics team leader, the Release Train Engineer presented the selection of metrics to the Scrum Masters. The meeting also served to inquire about the agile team acceptance and to secure the commitment of the Scrum Masters. The task for the Scrum Masters was to carry the metrics introduction project into the agile teams during the upcoming innovation and planning sprint. This two-week period at the end of every PI offers the Scrum Masters the freedom to implement ideas in workshops and discussions. The introduction of the metrics aims to enable the teams to self-direct, identify disruptions, and derive actions based on information. Furthermore, the metrics team wants to standardize the base of metrics for all teams with the perspective of introducing metrics at a scaled level. The Scrum of Scrums meeting participants plan to evaluate the metrics in the same joint format. For this purpose, every Scrum Master integrates its metrics into a central Jira dashboard. In the first step, the Scrum Masters collect the data within the agile teams and then migrate them to the central dashboard. Following the meeting, I created the measurement plan based on the planning for data collection and obtained validation from the RTE. Table 6.7 shows the object, scope, time/event, source, responsible, and means for all base metrics in the context of the measurement plan. Many of the base metrics relate to typical agile software development objects such as defects, different attributes of tests, refactors, the strengths and weaknesses of teams, various attributes of stories, the progress of sprint and PI, and the agile workflow overall. The number of stories postponed includes both stories pushed to the next sprint and those that were not accepted and deferred to a date later than the next sprint. The scope refers for most base metrics to the agile teams. Two metrics cover the entire ART at the program level. The point in time of the data collection is similar. Except for the two metrics, data is collected at each sprint. The remaining not completed epics per time unit are collected per PI. The cumulative flow diagram processes the data continuously. At the same time, participants also discussed the term for each metric, as shown in Table 8. Short-term means that the metric is collected from the beginning of the upcoming PI, and the Scrum Masters communicate the plan within the Innovation and Planning sprint. Medium-term metrics refer to an introduction till the end of the upcoming PI. Here, the RTE still needs to discuss details of the data collection with the responsible persons. Metrics with a long-term focus fall under the period after the upcoming

Goal	Question	Metric	Term
Increase the commitment to build trust and better plan work packages in terms of their content and scope from the perspective of customers and members within the ART	How committed is the sprint planning of the teams in terms of content and scope?	Number of added stories	Short
		Number of stories planned	Short
	How committed is the team to achieving the sprint planning?	Sprint burn-down chart	Medium
		Number of not accepted stories	Short
		Number of accepted stories	Short
		Percentage of accepted stories	Short
	How committed is the ART in achieving the PI planning?	PI burn-down chart	Medium
	How do disruptions affect commitment?	Number of stories pushed to next iteration	Short
		Number of stories not accepted and deferred to later date	Short
		Number of stories not accepted and deleted from backlog	Short
Improve the transparency of the status of workflows and requirements from the perspective of all members within the ART	How well do the workflows function within a PI?	Cumulative flow diagram	Long

Table 6.6.: Goal, Question, Metric Plan of the ART with introduction term: Part II

PI. Long-term metrics require a successful adoption of the other metrics and an improvement of the agile maturity level. The data source for many of the base metrics is Jira. The remaining metrics come from Tosca, the automated software testing tool, or from asking the teams and individual developers. The Scrum Masters of the respective teams, the test team, and the RTE share the responsibility depending on the metric. The ART captures most of the metrics in an automated fashion, although manual data collection must still be relied upon in some cases. After the meeting, I also shared the summary of success factors for introducing and using metrics with all participants at their explicit request. The success factors formulated as recommendations can be used by those responsible, first and foremost the Scrum Masters in their teams, to optimally shape the introduction.

Step 7: Collect, analyze and interpret the metrics data

After the measurement plan had been created in step 6, the ART metrics team decided to collect only the data of the short-term metrics in the first sprint of the new PI cycle. Therefore, only the short-term metrics are part of the observation in this step. Ensuring completeness, internal consistency, external consistency, the plausibility of data, and the separate handling of abnormal data is the responsibility of the Scrum Masters for the data from Jira and the test team for the data from Tosca. After the data was collected according to the responsibilities in the measurement plan via the tools defined therein, some base metrics still had to be aggregated. These metrics include the percentage of automated tests aggregated from the base metrics number of automated tests and the total number of tests. In addition, the metric percentage of accepted stories combines the ratio of the base metrics accepted stories and planned stories. Besides the short-term metrics, different medium and long-term metrics can be derived from the base metrics. For example, the long-term metric unit test coverage consists of the base metric of the lines of code for the unit tests and the software to be tested. Moreover, the auto-generated metrics PI burn-down chart (medium-term) and cumulative flow diagram (long-term) use the base metrics not completed epics per time and stories per state. The short-term aggregated metrics were part of the analysis of the first sprint in the new PI as the short-term metrics that do not require any additional aggregation. The Scrum Masters and the RTE analyzed and interpreted the metrics within the Scrum of Scrums meeting after the first sprint. Especially in terms of commitment, the associated metrics help track the goal. It turned out that without the introduction of metrics, teams are not able to inform stakeholders. The three associated metrics also answered the question regarding the impact of disruptions. The same applies to the questions regarding the commitment in content and scope and the commitment in achieving the sprint planning. The metric results also showed that some teams have problems with over-planning. In the past, the teams were able to keep silent about these problems because no metrics were collected. The introduction of the metrics helps to prevent oblivion. In addition, the Scrum Masters attest that the introduced metrics form a solid basis for argumentation and make recurring problems transparent. The RTE argues that the metrics team can now uncover systematic deviations via the chronological listing in the dashboard. The Scrum Masters currently still manually enter the metrics data from Jira collected for each sprint into the Confluence dashboard. The

Base metric	**Object**	**Scope**	**Time/event**	**Source**	**Responsible**	**Means**
Number of defects	Defects	Agile teams	Sprint	Jira	Scrum Master	Automatic
Lines of code	Unit tests & software	Agile teams	Sprint	Tosca	Test team	Automatic
Number of new test cases	New test cases	Agile teams	Sprint	Developers	Scrum Master	Manual
Total tests	All tests	Agile teams	Sprint	Tosca	Test team	Automatic
Number of automated tests	Automated tests	Agile teams	Sprint	Tosca	Test team	Manual
Number of new automated test cases	New automated test cases	Agile teams	Sprint	Tosca	Test team	Manual
Number of refactors	Refactors	Agile teams	Sprint	Developers	Scrum Master	Manual
Team Self-Assessment	Strengths & weaknesses	Agile teams	Sprint	Agile team	Scrum Master	Manual
Number of added stories	Added stories	Agile teams	Sprint	Jira	Scrum Master	Automatic
Number of stories planned	Planned stories	Agile teams	Sprint	Jira	Scrum Master	Automatic
Not completed story points per time	Sprint progress	Agile teams	Sprint	Jira	Scrum Master	Automatic
Number of not accepted stories	Not accepted stories	Agile teams	Sprint	Jira	Scrum Master	Automatic
Number of accepted stories	Accepted stories	Agile teams	Sprint	Jira	Scrum Master	Automatic
Not completed epics per time	PI progress	ART	PI	Jira	RTE	Automatic
Number of postponed stories	Postponed stories	Agile teams	Sprint	Jira	Scrum Master	Automatic
Number of deleted stories	Deleted stories	Agile teams	Sprint	Jira	Scrum Master	Automatic
Stories per state	Agile workflow	ART	Continuously	Jira	RTE	Automatic

Table 6.7.: Measurement plan of the ART

dashboard consists of a table that contains all short-term metrics of all Scrum teams. This central dashboard ensures standardized metrics reporting and allows the metrics team to discuss possible interventions evidence-based in the Scrum of Scrums meeting.

Step 8: Check the achievement and coverage of goals regularly

A short-term review of the goal coverage takes place in the context of the Scrum of Scrums meeting, similar to step 7. The participants meet every two weeks after a sprint. During this meeting, the Scrum Masters of all agile teams can propose suggestions to the Release Train Engineer regarding the use of metrics. In addition, the participants observe together whether the metrics values answer their associated questions and contribute to the goal achievement or not. The Release Train Engineer also transfers far-reaching results discussed in this meeting to the steering committee. The steering committee can initiate impulses for a re-prioritization of the goals or extended information requirements. The steering committee meets every week. The participants discuss the topic of metrics and the associated goal coverage only if there is demand.

6.2.2. Improvement of the measurement program management

Step 1: Assess the metrics team maturity level

The metrics team maturity assessment took place as a remote interview. The Release Train Engineer, who is responsible for the topic of metrics, and one of the Scrum Masters participated to represent the perspective of the team level. In the one-hour interview, the participants rated the statements of the MeTeaM template in a dialog based on the four levels of agreement. The participants took turns reading out the statements and giving their assessment on behalf of the metrics team. In case of uncertainty or contradiction, the second person intervened, and the assessment of the particular statement was discussed in more detail.

Step 2: Evaluate the metrics team maturity level

The measurement program and the metrics team are at the beginning of their development. It has been less than a year since the introduction of metrics at a scaled level. The understanding of the benefits of metrics is just beginning to develop. The result of the maturity assessment of the ART metrics team in Figure 6.2 demonstrates the existing potential for professionalization. The radar diagram consists of one line showing the calculated percentages for the six different areas. In total, 17 percent of the responses fall into the fully compliant response option. The interview participants answered the majority of statements as partially compliant, with 42 percent. Furthermore, they rated 29 percent of the statements as missing. 13 percent of the statements were not (yet) relevant to them. A look at the radar diagram makes it apparent that all areas except for the area of team performance are below 40 percent. In the area of set-up, the metrics team is 25 percent compliant with the statements. Participants rated seven statements each as partially compliant or missing. They consider statement 1 on the formal set-up of the metrics team and statement 13 on the measurement sponsor to be irrelevant.

The first sub-area of set-up, formalization, has a score of only ten percent. The sub-areas roles in the team and roles in the organization are rated better with 25 percent and 50 percent confirmation, respectively. The second area, commitment, scored better overall at 38 percent. By participating in this research project, statement 21 is the only one in this area to be fully compliant. The metrics team is partially compliant with four statements, and three are missing. Statement number 24 regarding explicit management support for research received a rating of not relevant by the participants. The percentages for the team commitment and the organization's commitment sub-areas differ. Team commitment has a maturity level of 50 percent, while the organization's commitment reaches a value of 17 percent. In terms of impact, the maturity level of the metrics team stands at 32 percent. The participants assessed the metrics team regarding the usage of metrics deliverables in the organization (statement 27) being completely compliant. Five times each, they rated statements as partially compliant or missing. The impact sub-area outside/outreach contains the statements about participation in conferences, standardization committees, and the organization of hackathons that are not relevant for the metrics team. This subarea achieves a total of 25 percent agreement. In the other impact subarea, on the organization, scores the metrics team 33 percent. The fourth area, longevity, shows a maturity level of 29 percent. Four of the statements are partially compliant. Three statements lack any agreement. According to the interviewees, statement 41 about surviving organizational change and statement 47 about career opportunities are not relevant. The current efficiency/work practices of the metrics team meet the statements 25 percent of the time. There is a partial agreement in two statements and no agreement at all in two other cases. Statement 50, where it is about a short lead time for introducing new actions and information products, does not play a role. The final area of team performance comes in at 74 percent overall. The metrics team already fulfills ten statements. In the case of eight statements are only some parts missing to be fully compliant. All statements are relevant to the team in this area. The team does not yet fulfill statement 54 in the context subarea on the common understanding of the key internal and external stakeholders. As a result, the score for this subarea is 25 percent. In the sub-areas mission, talent, and buy-in, the team fulfills the statements at 50 percent. In the norms subarea, the percentage is 88 percent. The resources and courage subareas perform with 100 percent compliance. Tables A.7, A.8, A.9, A.10, A.11, and A.12 of Appendix A.3 provide an overview of the ART metrics team's detailed agreement behavior for the 72 statements by area.

Step 3: Plan and implement actions to improve the maturity level

The metrics team identifies room for improvement in all areas except team performance. Nevertheless, it would like to focus on the areas of set-up and commitment first. In particular, the formal set-up of the metrics program should be reinforced. That is why, in the formalization sub-area, the team has set its sights on two previously non-compliant statements, 2 and 4. Statement 2 states that the metrics team has a defined assignment, which reflects in the line manager's approved purpose and set-up of the team. Going forward, the metrics team accepts the metrics introduction assignment and tries to fulfill its purpose. The ART recognizes the purpose of the metrics team, even if the approval of a line manager remains

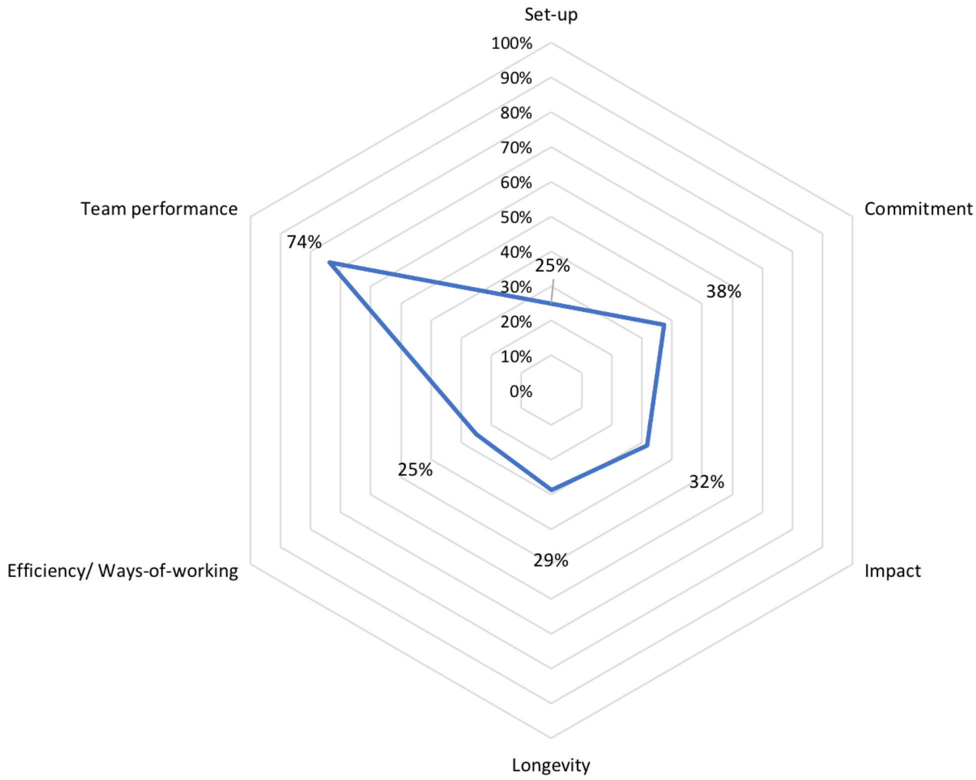

Figure 6.2.: Radar diagram of the ART metrics team maturity assessment

irrelevant to the team. The fourth statement stipulates that the metrics team defines its ways of working in the form of documentation. In preparation for the upcoming PI, the metrics team already started documenting the metrics introduction approach in the Scrum of Scrums meeting in Confluence. In the future, it intends to continue documenting the work practices in Confluence and, thus, fulfill statement number 4. The metrics team also plans to improve in the sub-area of roles in the team. For this purpose, the Release Train Engineer assumes the role of metrics team leader. For the thesis research project, the RTE already acts as the central contact. At the same time, the RTE also takes additional responsibility for specifying what is measured, when, and how. Hence, the metrics team increasingly complies with statement 7 having a metrics team leader and statement 14 having a measurement process owner. In the area of commitment, the metrics team is seeking advancements for both the teams and the organization. The first statement that is missing in the implementation of the commitment area so far is number 17. This statement includes the continuous improvement of the ways of working of the metrics team. In the future, the team intends to pay attention to this point and wants to improve its ways of working. The started metrics introduction can be regarded as the first expression of the improvement in this direction. The second statement in the organization's commitment sub-area insists on creating a metrics community as a community of practice within the organization. The metrics team plans to realize statement 25 by establishing a community of practice within the regular Scrum of Scrums set-up. The exchange within the newly founded metrics community will occur two or three times per PI cycle with all Scrum Masters and the metrics team leader. Even though the metrics team is not actively aiming at any other areas besides the two mentioned set-up and commitment, it still expects to improve further statements. These statements include, for example, statement 35, which emphasizes the significant positive impact of the metrics team's work on the customer's/organization's business. In the future, the metrics team will comply with this statement by supporting the organization with numbers and deriving appropriate actions to deliver faster and better functionality for the customer. The second example comes from the area of efficiency. Statement 51 is about the metrics team's ability to easily add, remove, change, and evolve actions and information products. The metrics team expects that the upcoming introduction of metrics results in skill development and a learning effect on this point.

Step 4: Perform the metrics team maturity assessment regularly

The metrics team would like to perform the maturity assessment regularly every six months, i.e., every two PI cycles once. During this time, the metrics team has enough time to implement the findings. The metrics team leader can imagine a reassessment after the upcoming PI to get fast feedback for the implemented actions. In summary, the metrics team expects that the experience gained through the periodic application of the assessment helps to improve the measurement program of the ART continuously.

7. Evaluation

This chapter contains the final evaluation of the developed artifact in the context of this thesis. This evaluation draws on two surveys that ask about the metrics teams' experience with the two different parts of the approach. The surveys aim to describe and quantify the approach with its steps and components in terms of their quality. Moreover, the surveys try to capture further refinements to improve the developed solution artifact. The survey method leverages a fixed research design within the case study environment [95]. According to Peffers et al.'s design science method, the evaluation corresponds to the fifth activity and, thus, decides "how well the artifact supports a solution to the problem" [86]. To find out the problem-solving competence of the artifact, I presented the theoretical structure of the approach to all participants already during the interview series. Many of the participants then actively carried out the steps of the approach. Only a few did not directly participate in the maturity assessment, which meant they were not invited to complete the corresponding survey. Overall, I sent the survey covering the goal-based metrics adoption part of the approach to all ten interview participants. The ten interview participants with their roles and experience in agile software development are shown in Table 4.1. All ten interviewees also completed the survey. The survey that concerns the part of the approach dealing with the improvement of metrics management includes seven of the ten interview participants. Three were explicitly excluded beforehand. This group refers to the Sales ART and comprises the Product Manager, a Scrum Master, and the developer. Of the remaining interview participants, all seven responded. Thus, seventeen of the planned surveys were completed, corresponding to an overall response rate of 100 percent.

Section 7.1 shows the evaluation for the first part of the approach. Section 7.2 evaluates the results for the second part of the approach. Both sections go through the questions of the surveys one after the other. Except for the open-ended questions for suggestions for improvement, the survey questions have end-verbalized five-point scales. These five-point scales provide survey participants sufficient differentiation regarding their opinions about the approach. A collected overview of all questions offers Appendix A.4.

7.1. Evaluation of the goal-based metrics introduction

This section presents the evaluation results for the first part of the solution artifact. In addition, the results of the survey questions on the goal-based introduction of metrics are brought into relation with the extant literature. Each subsection addresses a group of questions as posed by the survey, usually with an associated statement.

7.1.1. Stakeholders involved in the approach

Statement: "The approach should engage everyone involved in planning, defining, maintaining, operating, and managing metrics, if possible, to increase awareness and adoption of metrics."

Question 1: To what extent do you agree with the statement?

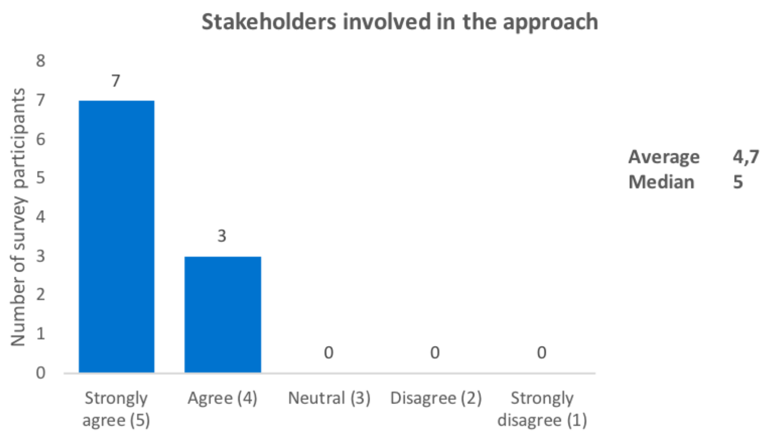

Figure 7.1.: Evaluation results for question 1

The result of the question in Figure 7.1 strongly supports the statement, and thus the view underlying the approach, that everyone who deals with metrics should be involved in adopting and managing of metrics. This finding is consistent with Staron and Medig, who emphasize the commitment of a dedicated measurement team and the teams being measured for metrics success [104]. The findings also follow up on Fagerholm et al. [34], Horlach et al. [47], and Samios' [96] recommendation to actively engage team members in the metrics and goals definition process to strengthen their understanding of purpose.

7.1.2. Evaluation of steps 1 and 2

Statement: "The approach starts with an as-is analysis of the already existing metrics and their goal coverage. In this way, potentials for the introduction of metrics based on goals can be identified."

Question 2: How valuable do you think the as-is analysis is as a starting point for the approach?

The as-is analysis is an extension of the generic GQM method. Steps one and two were thus incorporated into an approach and tested for the first time. Figure 7.2 shows that an overwhelming majority of 90 percent of the survey participants consider these two steps to be very valuable. The result suggests that every metric introduction project should first conduct an as-is analysis of the metrics in use and the goals they cover.

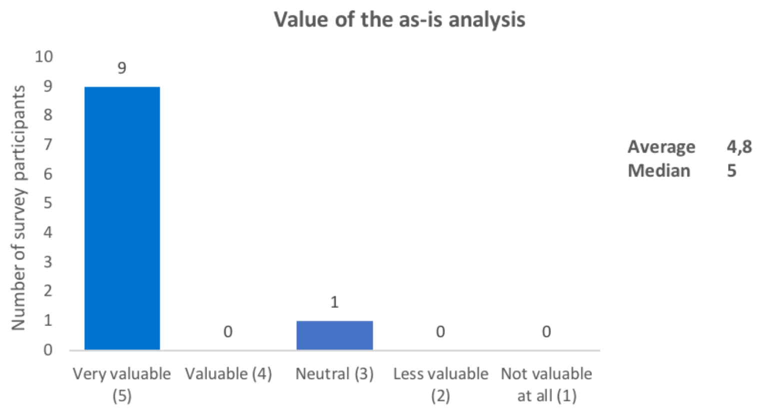

Figure 7.2.: Evaluation results for question 2

7.1.3. Evaluation of step 3

Statement: "The approach envisages collecting and prioritizing the agile software development goals of all persons responsible for this. The metrics team should include the as-is analysis of the already existing metrics and their goal coverage. If there are too many goals, the approach recommends limiting the number of goals to keep the metrics candidate list manageable."

Question 3: How valuable do you think is the goal-based metrics adoption?

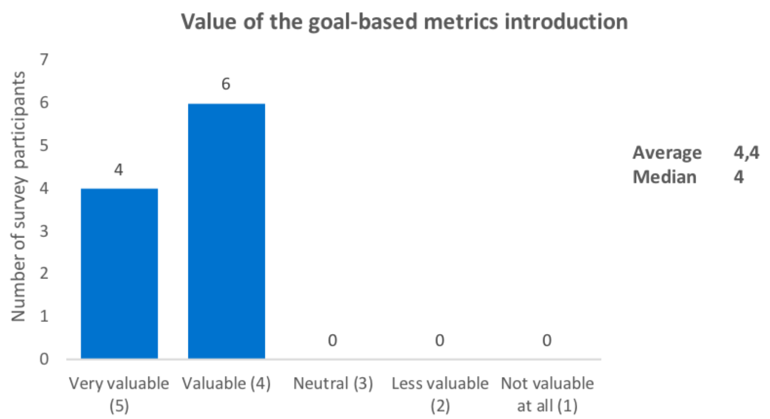

Figure 7.3.: Evaluation results for question 3

Many authors in the field of agile metrics research have demonstrated that goal-based metrics adoption is a key success factor. This is underlined not only by the work of Kettunen et al. [55], Horlach et al. [47], and Korpivaara et al. [58] but also by the increased use of the GQM method in agile software development measurement. The presented approach has made this success factor its brand core. The evaluation in Figure 7.3 confirms the value of this design in the scaled environment, where there are significantly far-reaching and overlapping goals. All ten survey participants see the goal-based introduction of metrics as valuable or very valuable.

Question 4: How useful do you think is the prioritization and recommended limitation of goals?

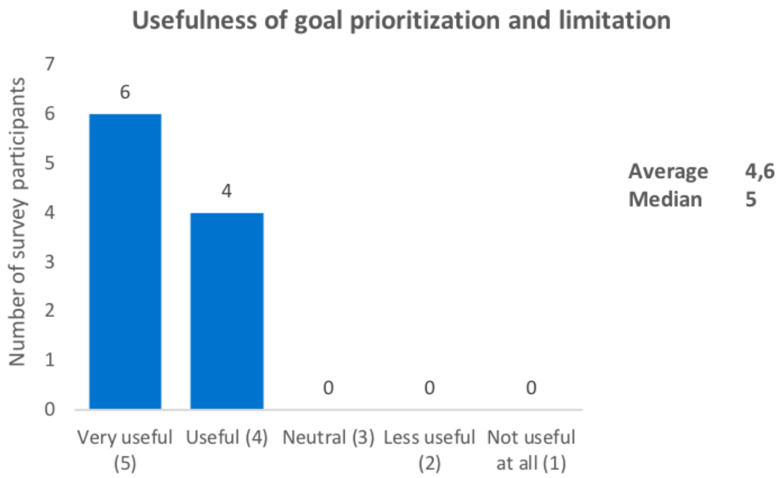

Figure 7.4.: Evaluation results for question 4

Step 3 of the approach recommends prioritizing and constraining goals for practical reasons. This feature has emerged through applying the approach and distinguishes the approach from, for example, the GQM method. Survey participants consider this feature useful or very useful (see Figure 7.4), thus advocating for its retention.

Question 5: How helpful do you think is the inclusion of the goal coverage in prioritizing the goals?

Two persons consider the inclusion of the goal coverage in goal prioritization as neutral or less useful (see Figure 7.5). This perception could be due to the fact that goal prioritization is feasible without the analysis of the goal coverage. Nevertheless, 80 percent think that inclusion is (very) helpful.

7.1.4. Evaluation of step 4

Statement: "The approach assigns at least one question to each goal. The questions should help to operationalize the goals and express the information interest regarding the metrics. The metrics should end up answering the question if possible. A hypothesis can also be added to each question to compare the expected outcome with the measured outcome."

Question 6: How useful are the questions as a link between goals and metrics?

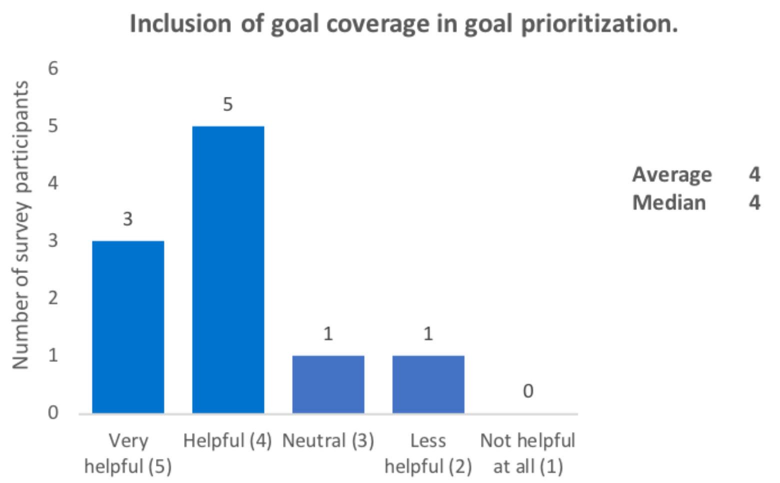

Figure 7.5.: Evaluation results for question 5

The great majority of survey participants rate the questions linking goals and metrics as helpful (see Figure 7.6). Hence, they confirm the step suggested by the GQM method.

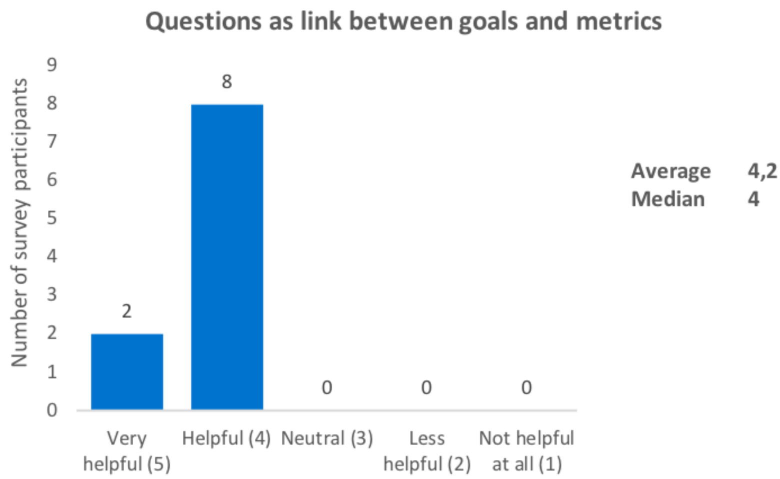

Figure 7.6.: Evaluation results for question 6

> **Question 7:** How valuable is the hypothesis for achieving a learning effect by comparing expected and measured outcome?

Even fifty percent of the participants each regard the hypothesis associated with the question as very valuable and valuable (see Figure 7.7). Thus both the questions and the hypotheses are validated as components of the fourth step of the approach.

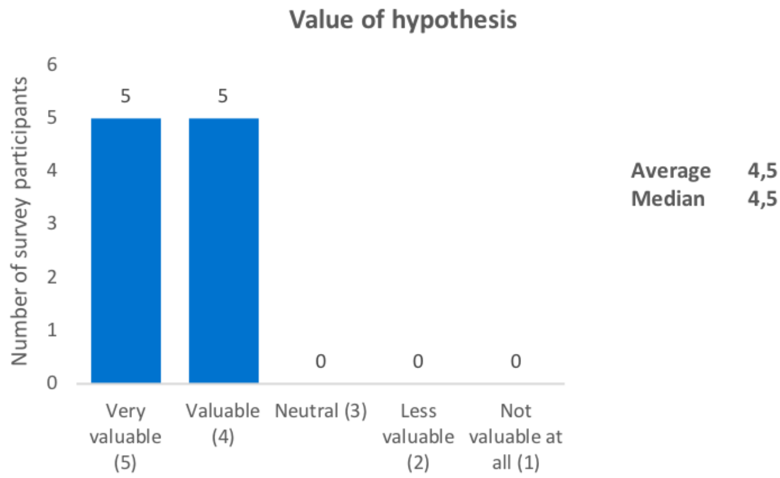

Figure 7.7.: Evaluation results for question 7

7.1.5. Evaluation of step 5

Statement: "The approach looks for metric candidates that cover the respective goals and answer the questions on a quantitative level. The metrics catalog with the goal-metrics mapping and the agile frameworks with concrete proposals for metrics serve as aids. There is also the possibility to adapt the metric candidates or to define new metrics. Then one selects the desired metrics from the candidate list."

Question 8: How useful do you think it is to search for candidate metrics and select specific metrics?

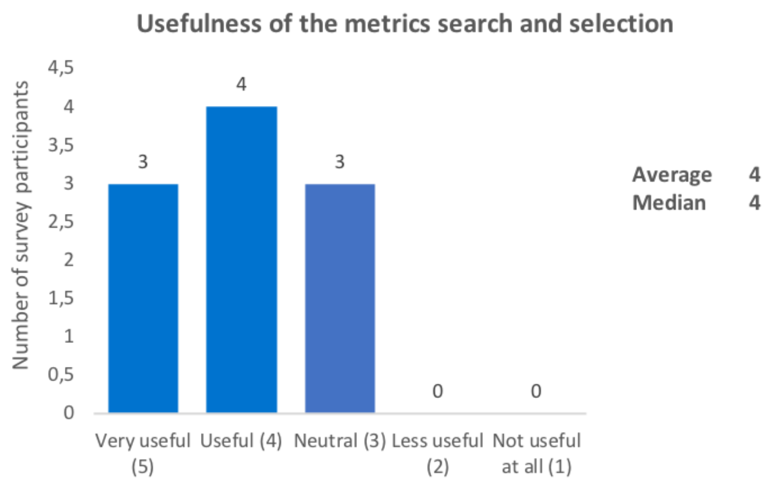

Figure 7.8.: Evaluation results for question 8

The usefulness of step 5 with its metrics search and selection are rated by three participants as neutral, four find them useful, and three rank them as very useful. Figure 7.8 shows that, on average, participants consider the step valuable overall. Of course, the step is essential for the approach because metrics could not be introduced without it.

Question 9: How helpful is the metrics catalog in searching and selecting metrics as part of the approach?

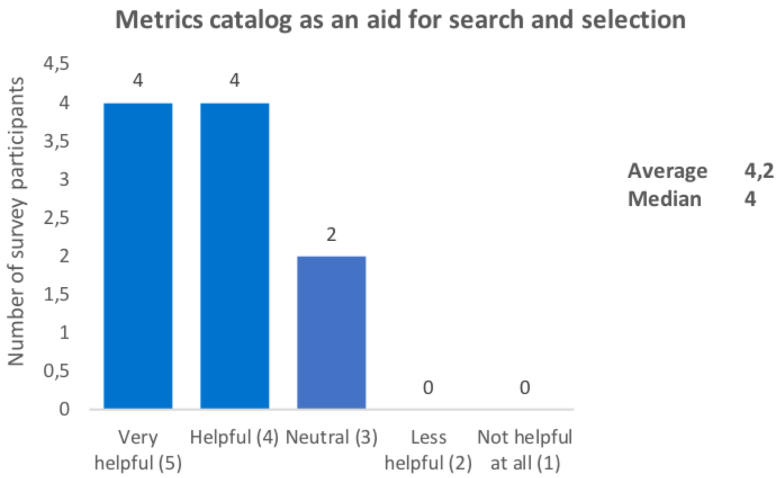

Figure 7.9.: Evaluation results for question 9

Figure 7.9 demonstrates that 80 percent of participants are satisfied or very satisfied with using the metrics catalog in searching for and selecting metrics. Two participants, however, rated the metrics catalog as neutral in its ability to help. Nonetheless, the results validate the practicality of the metrics catalog.

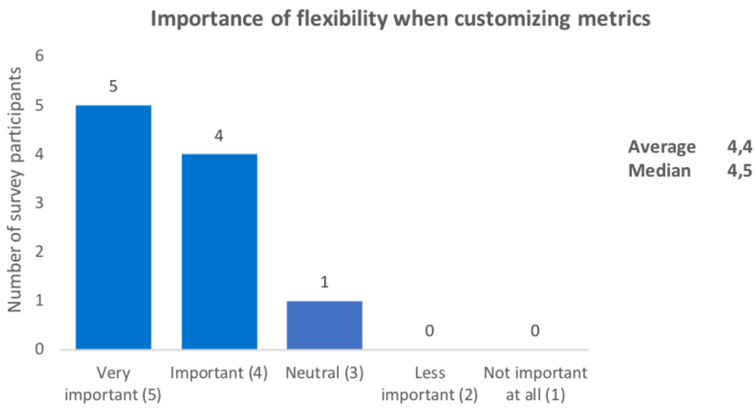

Figure 7.10.: Evaluation results for question 10

Question 10: How important is the flexibility to adapt and redefine metrics?

Fifty percent of survey participants consider flexibility in customizing and redefining metrics to be very important, according to Figure 7.10. This result is in line with the findings of the study by Coyle and Barata [22]. The two researchers identified an increase in team engagement due to flexibility. However, it should be noted that this flexibility should not come at the expense of a shared standard at an organizational level in the scaled environment.

7.1.6. Evaluation of step 6

Statement: "After selecting the metrics, the preparation of the data collection follows according to the approach. For this purpose, a measurement plan is created that defines the object, scope, time or event, data source, responsibility, and the means for each metrics measurement."

Question 11: How important is the step of preparing the data collection?

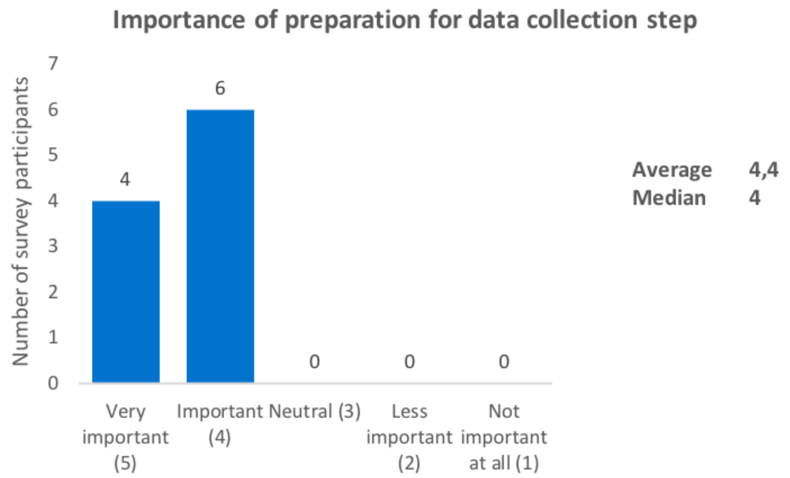

Figure 7.11.: Evaluation results for question 11

Step 6 is perceived as (very) important by the survey participants, as Figure 7.11 shows. The GQM method also included this step as a generic phase. The results indicate that the step is particularly important in the context of scaled agile development with different data sources, responsibilities, and divergent calculation techniques.

Question 12: How helpful do you consider the measurement plan with its components?

Even though two participants are only neutral about the measurement plan with its components and cannot derive any direct help, the vast majority view it differently. Figure 7.12 shows that eight out of ten participants regard the plan as helpful or even very helpful. This aspect has not yet been explicitly mentioned as a success factor in the context of agile metrics, although Basili et al. [7] describe the structure of a measurement plan.

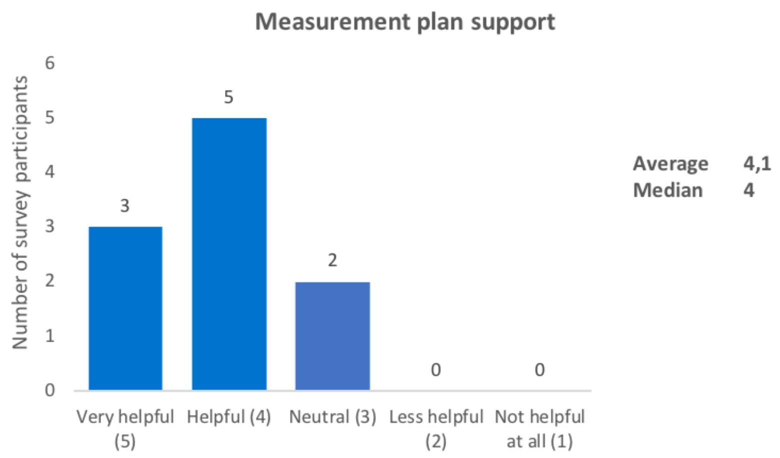

Figure 7.12.: Evaluation results for question 12

7.1.7. Evaluation of step 7

Statement: "The analysis plan next determines when, by whom, and how the metrics data will be analyzed. The values of the metrics are then interpreted in terms of answering the questions, testing the hypotheses, and achieving the goals."

Question 13: How important do you consider the analysis plan and the subsequent interpretation along the questions, hypotheses and goals?

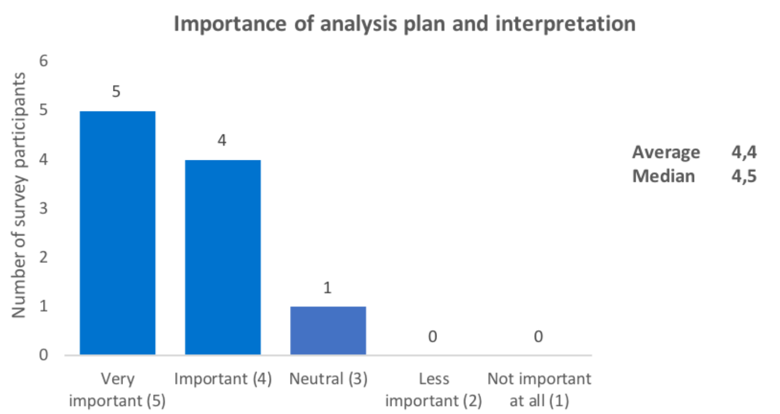

Figure 7.13.: Evaluation results for question 13

Step 7, with the definition of responsibilities in the analysis plan, data collection, evaluation, and interpretation, is rated by the majority of the survey participants as very important (see Figure 7.13). The GQM method also includes this phase. Due to the high importance, the step has also proven successful for this approach.

7.1.8. Evaluation of step 8

Statement: "The approach suggests a regular review of goal achievement and coverage because goals can change and, thus, metrics need to be adjusted as well. Goal coverage can also be insufficient, for example, so that new metrics have to be added, or existing metrics no longer add value. The review should be integrated into the iterative process of the agile framework."

Question 14: How valuable do you think is the regular review of the goal achievement and coverage?

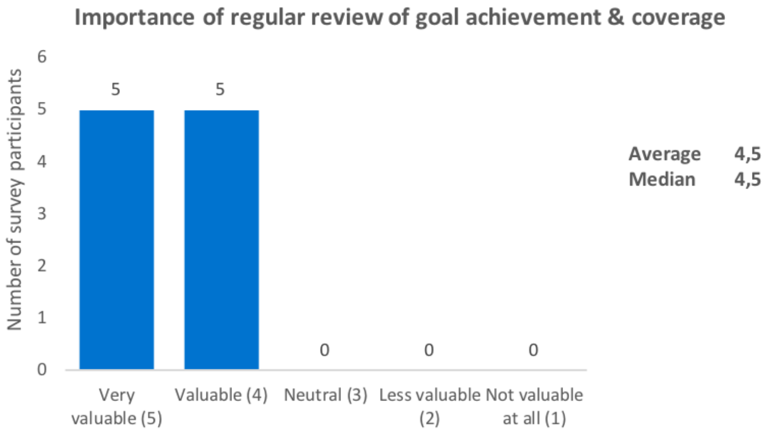

Figure 7.14.: Evaluation results for question 14

This aspect of the iterative review brings a novelty with itself and represents an extension to the GQM method by the approach. The approach takes up a factor that is important for the long-term success of measurement programs and incorporates it into the approach as a separate reiterative step. The importance of the step can be verified unambiguously by the survey result for this question, as shown in Figure 7.14. The result is thus in line with Ram et al. [91] and especially Korpivaara et al., who have already pointed out the importance in the context of scaled software development [58].

Question 15: How useful is the integration into the flow of the agile framework you are using in your company?

The results for the second question in Step 8 indicate predominantly the importance of integrating the approach into an existing agile framework (see Figure 7.15). One survey participant is neutral towards the integration, while one person considers it less useful. Despite this one deviation, both average and median are within the range of the previous survey results. In any case, the integration represents an important innovation concerning the long-term adoption of metrics. The result can confirm its practical importance.

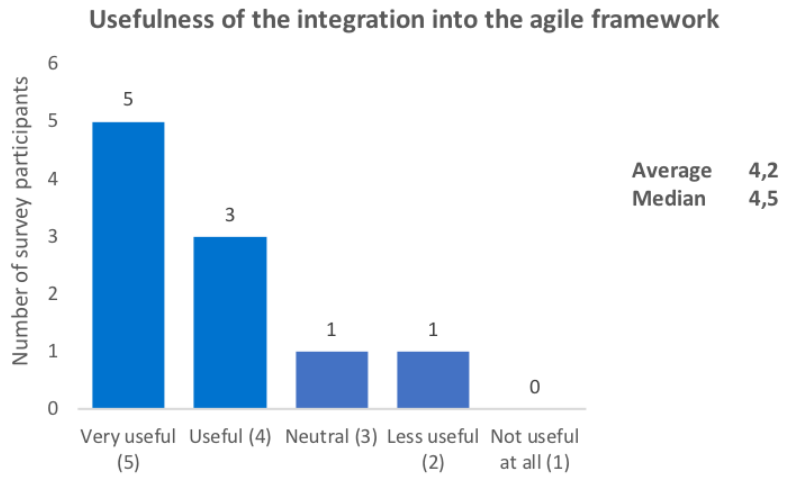

Figure 7.15.: Evaluation results for question 15

7.1.9. Overall evaluation of the approach

Question 16: How do you evaluate the goal-based metrics adoption approach overall?

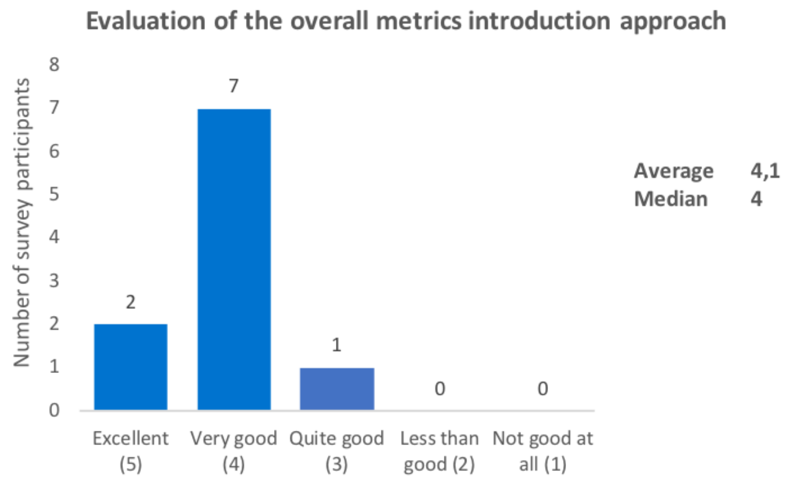

Figure 7.16.: Evaluation results for question 16

The results presented in Figure 7.16 explicitly prove the overall value of the approach. Twenty percent of the participants say that the approach is excellent. The majority, namely 70 percent, think that the approach is very good. One participant still agrees that the approach as a whole is quite good.

Statement: "I am convinced that the approach will help the company to select the right metrics and to make our goals verifiable."

Question 17: To what extent do you agree with the statement?

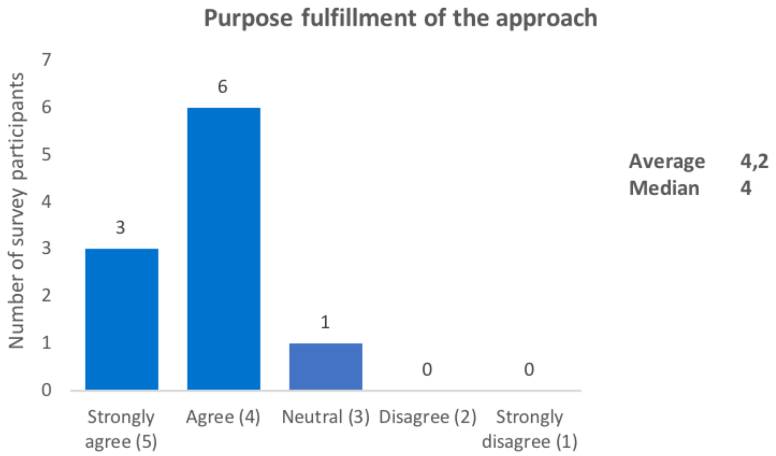

Figure 7.17.: Evaluation results for question 17

Following up on the result in the previous figure, survey participants agree on the fulfillment of purpose. Ninety percent of participants say that the approach helps them select metrics and meet goals (see Figure 7.17). The approach in its current form thus lives up to its claim to enable metrics teams in the context of large-scale agile software development to select metrics based on strategic goals through a systematic procedure. Such an approach, especially in the context mentioned, has not existed before.

To the three questions regarding improvements due to removing, changing, or adding steps in the approach, there were two suggestions: First, identify the group of people who communicate feedback on the introduction results in general and the metric values in particular to all teams involved. This task should fall to the metrics team leader, as this person has overall responsibility for the project and has a central overview. So far, this does not seem to have been done sufficiently in the execution of the approach. Second, the agile teams should define frameworks that specify under what conditions there should be changes in software development. This suggestion is in line with recommendation R18 and the observations of Cheng et al. [17].

7.2. Evaluation of the measurement program management improvement

Statement: "As part of the metrics team's improvement approach, the maturity model gave us an overview of the strengths and weaknesses of the current measurement program."

Question 18: To what extent do you agree with the statement?

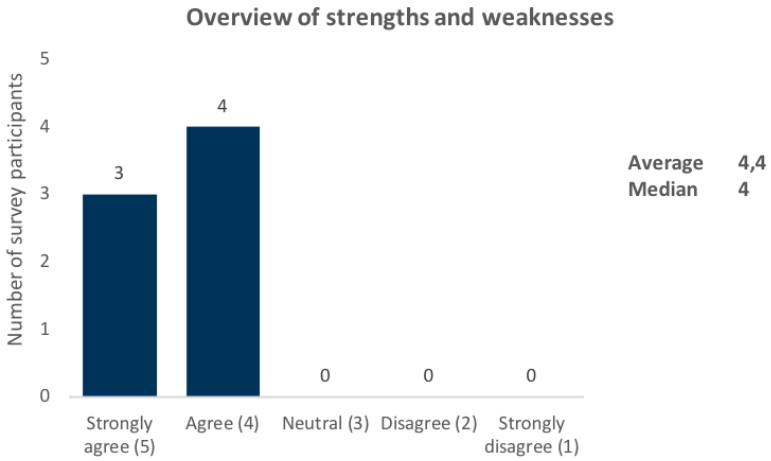

Figure 7.18.: Evaluation results for question 18

All survey participants either agree or strongly agree with the statement, as shown in Figure 7.18. This result confirms the conclusion of the evaluation by Meding et al. [73], which attributes to the MeTeaM model the ability to communicate strengths and weaknesses of the measurement program to teams.

Statement: "The maturity assessment is easy to perform."

Question 19: To what extent do you agree with the statement?

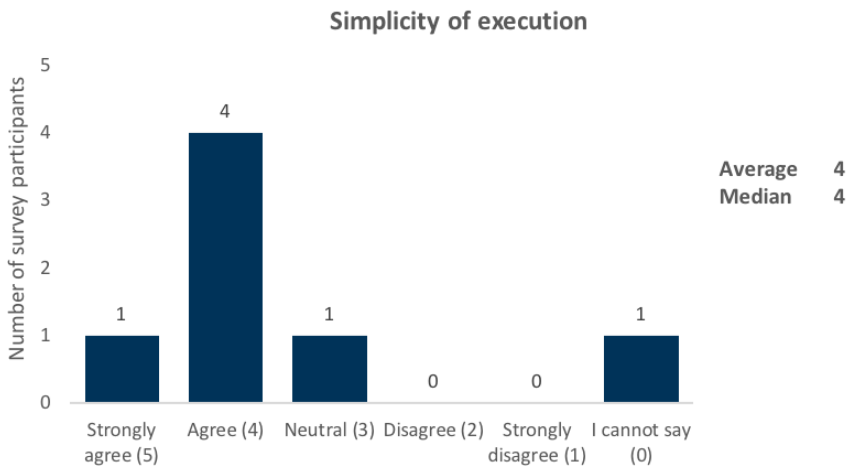

Figure 7.19.: Evaluation results for question 19

This aspect was a sticking point during execution because the template contains over 70 statements. Nevertheless, the majority of participants felt that it was easy to conduct the assessment. However, as shown in Figure 7.19, there is also a neutral position on this statement. Regarding a possible improvement of a step, one participant stated that it would be helpful to limit the number of statements depending on the maturity level. In his opinion, "some dimensions are only relevant at a certain level of maturity." This suggestion would mean that the model, which comprehensively queries all possible aspects of metrics team maturity, would need a refinement that categorizes statements and dimensions according to the maturity level. The task would then be to find out which statements and dimensions are relevant at which maturity level. This thesis has found that the areas of set-up and commitment are particularly relevant to teams with lower levels of maturity. The area of longevity, in contrast, was not a starting point for action in these cases.

> *Statement:* "The maturity assessment covers the relevant and essential characteristics of a successful metrics team."
>
> **Question 20:** To what extent do you agree with the statement?

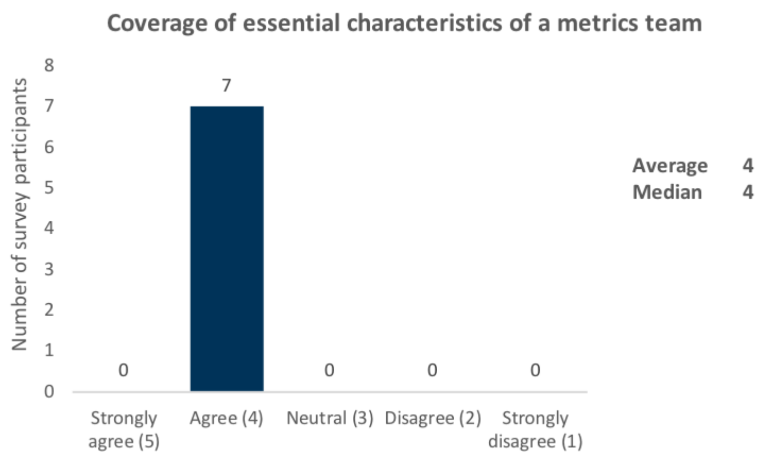

Figure 7.20.: Evaluation results for question 20

The survey participants unanimously agreed with this statement (see Figure 7.20). Refinement in this respect would rather be the limitation of the scope of the template and a stronger focus on smaller organizations. For example, it is unrealistic to assume that smaller organizations employ an entire team for measuring. Smaller organizations are more likely to distribute measurement tasks among several people without formally establishing a measurement team. At the same time, this is where the model has the potential to reflect on these deficiencies in metrics management.

> *Statement:* "The maturity assessment has revealed the potentials of the measurement program, so we now know what we should work on."
>
> **Question 21:** To what extent do you agree with the statement?

The result in Figure 7.21 speaks for itself and shows that the model's strength lies in uncovering the potential for actions that a metrics team can take to improve the management

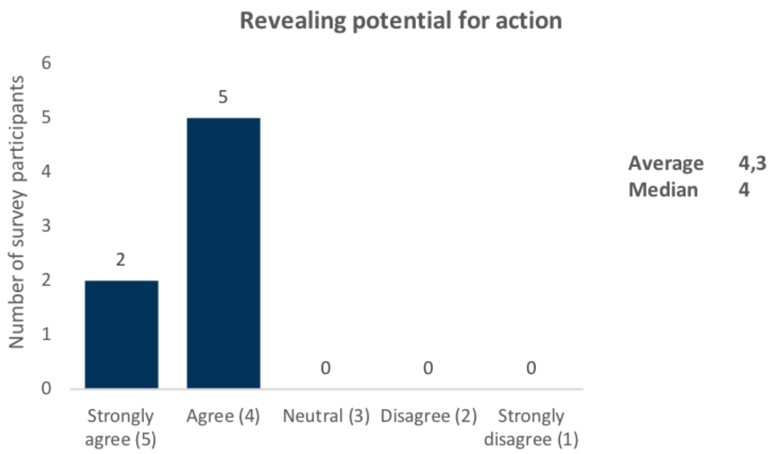

Figure 7.21.: Evaluation results for question 21

of the measurement program. The visualization and concise classification of the statements is also beneficial here.

> *Statement:* "The maturity assessment helps us professionalize the measurement program and the metrics team."
>
> **Question 22:** To what extent do you agree with the statement?

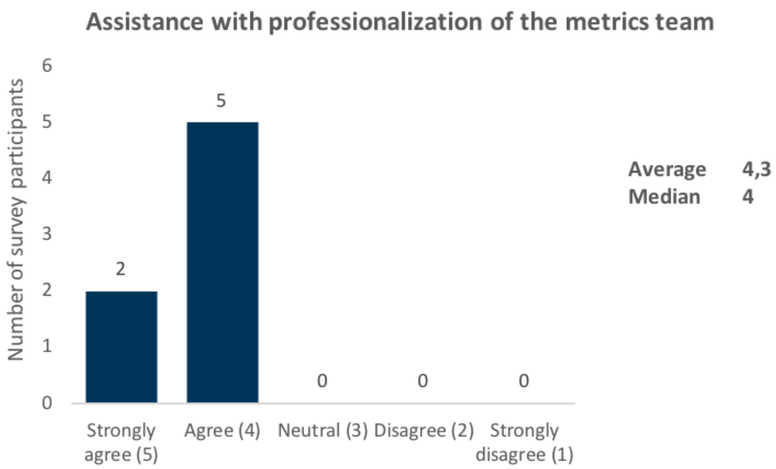

Figure 7.22.: Evaluation results for question 22

The same picture emerges when asked about support for the professionalization of the metrics team. Fulfilling this statement is a declared objective of the approach. Two participants strongly agree with the statement, and the remaining five agree (see Figure 7.22). This result proves that the approach achieves this objective at this point.

Statement: "We plan to conduct the maturity assessment at regular intervals in the future."

Question 23: To what extent do you agree with the statement?

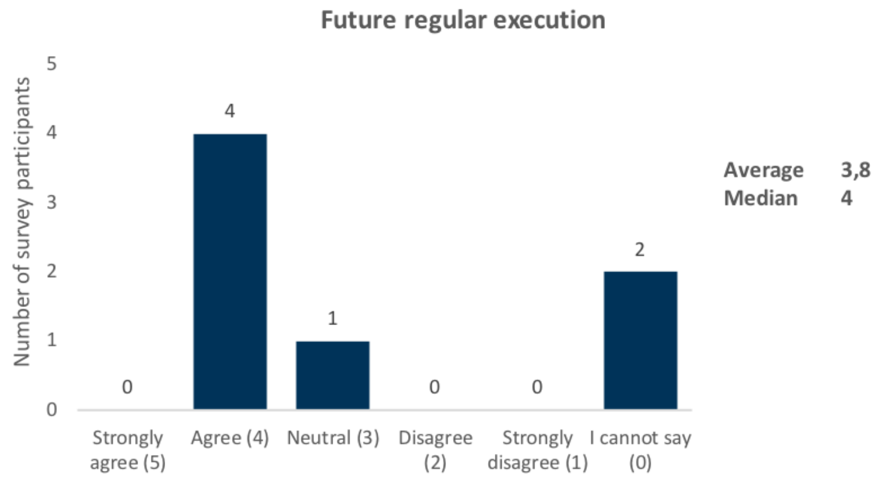

Figure 7.23.: Evaluation results for question 23

The level of agreement with this statement looks more restrained, as Figure 7.23 shows: Two participants abstain, one participant takes a neutral view, and four participants favor of a regular execution of the metrics team maturity assessment. This regular execution and review of the actions taken promises a steady and, thus, long-term improvement in the management of metrics programs.

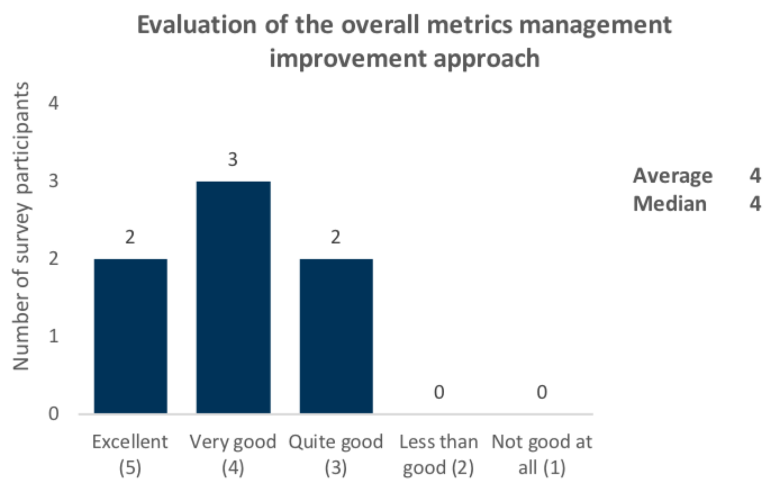

Figure 7.24.: Evaluation results for question 24

Question 24: What is your overall opinion of the measurement program management improvement approach?

The majority of the survey participants rate the final question on the overall assessment of the approach to improve the management of the measurement program as very good (see Figure 7.24). Two participants even see the approach as excellent, while two others see it as quite good. The result confirms the benefit of using the maturity model in the context of the approach of this thesis. Furthermore, the results validate the practicability of the model in a case environment outside the original one of Meding et al. [73].

8. Discussion

This chapter recapitulates the key findings of this thesis and discusses their possible causes and consequences. In addition, the limitations and shortcomings of the research are explained.

8.1. Key findings

The following section summarizes the relevant findings of the thesis in short paragraphs.

The current use of metrics in relation to goals in the scaled agile environment

The case study interviews revealed that the current use of metrics encompasses many diverse goals. According to Korpivaara et al.'s [58] classification, the goals are found at all organizational levels and cover various dimensions of it (see Table 4.2 and 4.3). Even if the decision-makers do not have explicit documentation, they can state their goals in the context of metrics. When it comes to the current use of metrics, the as-is analysis showed a very contrasting picture between the two analysis units. While the IT-provider's use of metrics is determined by the functional scope of Jira and limited to agile team metrics, many of the ART's metrics are calculated based on person-days at the program level. In addition, the use at the team level varies widely within the ART and, thus, does not follow a uniform standard. In both cases, the primary responsibility for metrics lies with those responsible for the Scrum or SAFe process: the Scrum Master at the team level and the Release Train Engineer or Scrum of Scrums Master at the program level. In addition to these roles, the Product Owners at the team level or Product Managers at the program level usually have an increased interest in metrics. They bear partial responsibility, especially for metrics relevant to planning. The responsibility for goal definition and compliance divides along with the areas of responsibility of the various agile roles. This means that the PO is responsible for the goals related to the product, such as product quality. The SM takes care of goals concerning the process, such as speed. The System Architect pursues technical goals. The management sets more strategic goals and delegates them to the SM or PO. Goals such as customer satisfaction are usually pursued collaboratively. Potential challenges in using metrics are the feeling of control, misinterpretation, horizontal alignment of metrics, and the limitation of autonomy and self-responsibility. These challenges coincide with existing studies [34, 39, 58, 83].

Identification of recommendations for the adoption of metrics in large-scale agile software development

Previous literature reviewed success factors for the introduction and use of metrics in various articles [83, 91, 104]. This thesis capitalized on these findings and transformed the success factors into concrete recommendations for metrics adoption. At the same time, the interviews contributed to confirming the already existing ones and developing new recommendations for the adoption in a scaled context. The resulting 27 recommendations were then integrated into the proposed approach and its steps, making them part of the solution artifact. Examples of new recommendations from the interviews include:

- Assigning responsibility for the introduction project to a central coordinating body (metrics team leader) [R12]
- Perceiving metrics introduction as an agile experiment [R6]
- Actively using metrics in cross-team agile events [R16]

Design and evaluation of an approach for the goal-based introduction of metrics in a scaled context

The thesis presents a novel approach to systematically introduce metrics based on goals in large-scale agile software development. The approach takes the aspects of goal-based definition and interpretation of metrics with questions as a link in between from the GQM method and integrates them into the scaled environment. The goal-based introduction as a central idea also proved (very) valuable in this environment (40 percent say very valuable and 60 percent valuable). The as-is analysis complements the original approach and lays the foundation for the steps that follow. The approach also calls for regular review of goals and metrics, which proved particularly valuable to survey participants in the evaluation. The integration into the large-scale agile environment accomplishes the approach through roles and events defined in the frameworks. Key stakeholders include Scrum Masters, Product Owners, and Release Train Engineers, who discuss the adoption in events such as the Scrum of Scrums or the Sprint Retrospective. Survey participants evaluate the approach as very good overall. This result shows the practical relevance of the approach. From a theoretical perspective, the approach extends the efforts of various scholars in single-team settings [13, 18, 22] and ties into the few efforts at a scaled level [58].

Design and evaluation of an approach that improves the measurement program management

The approach not only presents a way to introduce metrics but also offers a well-founded, holistic concept to manage metrics over the long term and evolve the measurement program. Therefore, the approach embraces the culture of continuous measurement, as Oza and Korkala [83] call for, and provides a reference framework for metrics management. Within the frame of reference, the application of Meding et al.'s metrics team maturity model enables the effectiveness and quality of a team's measurement program to be uncovered and action to be stimulated for improvement. The results of the evaluation of the approach show significant approval majorities for the capabilities to highlight strengths and weaknesses, reveal improvement actions, and professionalize the metrics team. Overall, more than 70 percent of survey participants agree that the approach is very good or even excellent. However, there are slight doubts about the regular application of the model in the future because the fear is that the necessary resources (people and time) are lacking. Some participants also suggested that the approach needs to be limited to the basic needs of smaller and less mature teams.

Findings from running the approach in two different instantiations

For the execution of the approach, it is essential to involve, if possible, all those engaged in planning, defining, maintaining, applying, and managing metrics in order to increase awareness and adoption of metrics. There are very high levels of agreement with this finding (see Figure 7.1). Team members of both instantiations prioritized the goals of quality and commitment very highly. The questions of the ART focus on test coverage, defects, test automation, achieving planning targets, and impact of disruptions. The IT-provider's information interest concerns feature quality, response time to critical defects, test automation, speed of development, and the proportion of effort spent on security. The ART introduced nineteen primarily agile metrics at the team level to answer the questions and satisfy the underlying information interest. Most of the metrics come from the SAFe Reference Guide and intend to standardize the metrics at the team level. The nineteen metrics have been and will be introduced in the short, medium, and long term. The IT-provider was guided in the selection of metrics by the idea of introducing cross-team metrics. The plan is to iteratively introduce the metrics over two months and present the result to management. The analysis and interpretation of the metrics at the program level of the ART take place in the Scrum of Scrums event. Here, the SMs exchange ideas with the RTE about possible actions. At the team level, the SMs discuss the metrics with the PO and the developers before the sprint in the Planning event and after the sprint in the Retrospective. The regular goal achievement review take over the SM, PO, and developers within the Sprint Retrospective meeting in the case of the IT-provider. The same stakeholders evaluate the metric values in the Sprint Review. New goals come from the Executive Meta Scrum event. The metrics team assessment at the ART uncovered potential in the areas of set-up and commitment. The team figured out concrete actions, such as assigning a metrics team leader or establishing a metrics community. Like the ART, the IT provider wants to improve the areas set-up and commitment and additionally identifies improvement potential for the metrics team's impact on the organization. In practice, both analysis units see a regular execution of the assessment every six months as realistic. Running the maturity model outside of the original research environment of Meding et al. [73] represents another novelty of this thesis.

First validation of the metrics catalog in practice

Even though Tobisch [111] evaluated the proposed metrics catalog via a survey, its practical use and validation have been lacking so far. As part of the execution of the approach in the dual instantiation, the metrics catalog came into use in two steps of the metrics introduction part. In Step 2, teams determined the goal coverage based on the previously identified metrics through bottom-up searches within the catalog. Teams applied the catalog's goal-metric mapping top-down in Step 5 to compile a list of metric candidates and then select the metrics that best matched the goals and questions. After execution, 80 percent of survey participants indicated that the metrics catalog was helpful or very helpful. The result proves the practical validity of the metrics catalog.

8.2. Limitations

This section discusses the limitations and threats to the validity of the thesis using the classification of aspects by Yin [117] and Runeson and Höst [95]. The aspect of construct validity represents the congruence between the operational measures studied and my understanding of the research project according to the research questions. When conducting the interviews, I used several interventions to precisely capture the current use of metrics and prevent respondent bias in any direction. For example, I explained various definitions such as large-scale agile or metrics to the interviewees. I also repeated the "what and how" of the approach at each interview before starting the questions. This also applies to the surveys, which contain much explanatory text to counter misunderstandings. Furthermore, the definitions, the presentation of the approach, and the explanatory texts refer to the theoretical framework, which describes the research construct thoroughly in chapters 2, 3, and 5. Indeed, the diversity of the interviewees' roles also contributed to the construct validity.

Internal validity aims at the quality of the research design, which should not allow any confounding variables to influence the measurement of causal relationships. Keeping the metrics team informed about progress in metrics adoption and maturity assessment results and limited research time counteracted history and maturation threats. In addition, I made sure, for example, in the maturity assessment that social interaction did not threaten the salience of responses.

The aspect of external validity addresses the possibility of generalizing results and determines the degree of interest from people outside the case under study. The execution of the approach will most likely differ from case to case, yet the description of the approach in Chapter 5 is as generic as possible and abstracts the various configurations of the two units of analysis. In general, the multi-case study design with the distinct team sizes and agile scaling frameworks of the analysis units contributes to external validity.

Reliability concerns the degree to which the data and the analysis depend on the researcher. Running the study multiple times using the same methodologies should yield the same results. To avoid compromising this aspect, I described the applied research approach and its methodologies in Section 1.3 and made the interview and survey questions transparent in Appendix A. However, repeating the research in a different organizational context might result in other metrics, goals, responsibility distributions, challenges, and recommendations because of different individual experiences, skills, roles or organizational structures, maturities, and cultures. Another limitation arises from the limited number of interview and survey participants. There is some threat to validity in the meaningfulness of the results. Moreover, the constrained time frame for the thesis limited the more detailed observation and evaluation of the long-term effects of the selected metrics and the targeted metrics management actions. Similarly, the scope did not allow for a more detailed review of the implementation of the metrics adoption recommendations.

9. Conclusion and future work

The final chapter of this master's thesis summarizes the main findings, answers the research questions, and situates the thesis within the state of research. In addition, the chapter presents an outlook for future research.

9.1. Summary

This master's thesis dealt with metrics in large-scale agile software development. To this end, the previous use of metrics in relation to strategic goals was first examined in a case study. Combined with proven concepts from the literature, these collected case study findings helped build and evaluate an artifact for adopting and managing metrics using the action design research method.

> *Research question 1:* How are metrics in relation to strategic goals currently being used in large-scale agile software development?

The case study results show that existing metrics are used implicitly in connection with goals, but the explicit, documented, systematic use is missing. The current use of metrics per se depends on the degree of adoption of agile ways of working and the scaled frameworks. Organizations with lower adoption levels are more likely to calculate metrics based on person-days. In contrast, organizations with established agile structures exhibit an almost exclusive use of story-point-based metrics. Similarly, the proportion of cross-team level metrics depends on the progress of adopting the scaled frameworks. The responsibility for introducing and using metrics in the large-scale agile environment lies with the Scrum of Scrum Master or Release Train Engineer as part of their process accountability. The responsibility of goal definition and tracking is shared according to the agile role distribution and is partially performed jointly. In addition, the case study has confirmed some key metrics challenges in a scaled context, such as the feeling of control, misinterpretations, or the horizontal alignment of metrics. These findings answer the first research question and tie in with previous research on large-scale agile software development metrics by Korpivaara et al. [58].

> *Research question 2:* How can different stakeholders establish and manage metrics and strategic goals to improve the current large-scale agile software development measurement program?

The approach proposed in this thesis demonstrates how metrics teams can systematically establish and continuously manage metrics in scaled agile software development. The core idea of the approach is the alignment of metrics with strategic goals, which builds on and extends previous studies on the GQM method [13, 18, 22]. The extensions include the as-is analysis of the metrics used and their associated goals, enabling a more precise selection of metrics and prioritization of goals. In addition, the approach describes the integration into the scaled agile framework via roles and events and suggests the regular review of goals and metrics. Also part of the approach are recommendations grounded in case study results and existing literature [83, 91, 104] for successful, long-term metrics adoption. Beyond that, the approach empowers metrics teams to improve the management of metrics programs in a holistic sense by assessing their maturity level. Surveys in the two case organization units indicate that the steps and components of the approach are useful, helpful, and valuable, revealing a high practical relevance. The survey results thus also confirm the value of Tobisch's metrics catalog [111], which the thesis validated for the first time after being used in practice. Overall, the execution of the approach enabled the two units of the case organization to introduce thirteen and nineteen different agile metrics, respectively, in an iterative, goal-based manner. Furthermore, the approach uncovered the potential of both metrics teams so that actions could be planned to improve their measurement program.

9.2. Future work

Accompanying and evaluating the introduction of the medium-term and long-term metrics in the Agile Release Train can be future work. The same applies to the iterative introduction in the case of the IT-provider. Here, it is a good idea to observe the adoption beyond the presentation to management and question the metrics team and the developers about the long-term success of the introduction. From this longer-term cooperation, refinements to the approach could emerge through further action design science iterations. This longer-term work could also include additional maturity assessments after three months to evaluate the continued development of the measurement programs and the success of the targeted actions. Similarly, it is reasonable to re-evaluate the approach after certain time through another survey run. In addition, future work could apply the approach and its re-evaluation in another case organization and adapt the approach as necessary to increase external validity.

A suggestion came from the case organization to equip the metrics with information on the degree of fulfillment of various goals. The degree of goal fulfillment would have the advantage of significantly simplifying and speeding up the metrics selection process because the level of alignment of goals and metrics would be more transparent and differentiated. Future work could explore the degree of alignment between metrics and goals. In addition, the dependencies between goals and their relative weights in the context of changing conditions in the agile environment could be an intriguing research direction. Currently, there is a lack of understanding of the adaption of metrics over time. There seem to be interrelationships and branching conditions among goals. The exploration of both phenomena requires longitudinal studies.

An extension of the approach is also conceivable in terms of digitizing the goal-metric mapping. Future work could develop an artifact with which teams no longer have to select metrics based on their goals in Excel or via a PDF catalog but can perform this step tool-supported. This artifact promises a significant gain in clarity and time savings. The prerequisite, however, is that each metric in the catalog has so-called "goal tags". As a result, all metrics would be discoverable when the metrics team enters its goals. During this thesis, the idea for an additional extension of the metrics catalog was born. This idea concerns the addition of standardized questions on the operationalization level between the existing goals and metrics. The questions within this thesis could serve as a starting point for this future work.

A. Appendix

A.1. Semi-structured case study interviews

A.1.1. General questions about the company and the role of the interviewee

1. What is your large-scale agile development (stakeholder) role?
2. How long do work in this role?
3. How long have you been working in agile software development?
4. How long have you been working in large-scale agile software development?
5. How long has your company been working with agile software development?
6. How long has your company been working with large-scale agile software development?

A.1.2. Questions about the usage of metrics in general

1. Which scaling agile frameworks are you using during the software development process?
2. To which level do you scale agile practices?
3. Which metrics do you use in large-scale agile development?
4. What information needs do you have that are not being met by the existing metrics?
5. Do you use any specialized tools to keep track of metrics and if yes, what are these?

A.1.3. Recurring questions about the most important metrics

1. What are the organization-specific goals, based on which the metric has been defined?
2. What is the metric related data and where is it collected from?
3. What is the calculation rule of the metric?
4. What are the metric's target, planned and tolerance values?
5. Which advantages do you perceive in using this metric?
6. Which disadvantages do you perceive in using this metric?

7. Which challenges did you face when implementing the metric?
8. What can be improved when implementing metrics?
9. Which challenges do you face when using this metric?
10. What can be improved in the use of metrics?
11. Based on the provided challenges, could you prioritize these starting from the most critical one?
12. How do you approach these challenges?
13. Which predefined countermeasures are you using?
14. Who is responsible for the definition of the metric?
15. Who is responsible for the selection of the metric?
16. Who is responsible for the implementation of the metric?
17. Who is responsible for the deletion of the metric?
18. How often and by whom is the corresponding metric being monitored?
19. Who are the persons in charge of the achievement of the predefined metric values?
20. Which predefined actions are used when the measurements deviate from the predefined metric values?
21. Which other stakeholders in your organization are interested in using the metric?
22. What is the reason for each stakeholder's interest in using the metric?
23. Which other metrics is the corresponding one related to?

A.1.4. Questions about goal setting, prioritization, and accountability

1. What are the company's strategic goals with regard to the production of software?
2. What goals exist related to the agile software metrics used?
3. Which goals are not covered by the metrics used so far?
4. Who is responsible for goal setting?
5. Who is responsible for meeting the goals?
6. How do you prioritize the various goals?

A.1.5. Recurring questions for goal concretization according to the GQM scheme

1. What is your purpose for the goal?
2. What is the problem that is solved by the goal?
3. Which object or quality attribute does the goal concern?
4. From what point of view should the goal be achieved?
5. What is the context of the goal?
6. Who is responsible for achieving the goal?
7. What questions would you need to ask to satisfy your demand for information regarding the goal?

A.2. Statements of the metric team maturity model

A.2.1. Team maturity area — Set-up

Sub-area	No.	Statement	Level
Formalization	1	The metrics team set-up is formal, e.g., the team is officially recognized by the organization	Org.
	2	The metrics team has a defined assignment, e.g., there is a purpose and set-up of the team, approved by a Line Manager	Org.
	3	The metrics team has dedicated resources, e.g., team members work full- or part-time in the team, and the organization knows who belongs to the team	Org.
	4	The metrics team has defined ways-of-working, e.g., the ways-of-working are documented	Team
	5	The metrics team has an approved definition of its vision, mission, and strategic plan	Org.
	6	There is a formal point of contact with the metrics team, e.g., JIRA	Team
Roles in the team	7	Metrics team leader: manages the team and ascertains the adherence to the assignment from the organization	Team
	8	Measurement designer: designs, implements and tests measurement systems	Team
	9	Measurement analyst: responsible for planning, execution, evaluation and improvement of measurement specifications	Team

	10	Database administrator: responsible for managing data store(s), definition of the architecture of the measurement infrastructure, automation of access to data sources, and monitoring of the information quality	Team
	11	Data analyst: conducts analysis on data and interacts with stakeholders and measurement users	Org.
	12	Data scientist: helps stakeholders to interpret data, performs predictions, leads AI activities, and supports management in taking data-based decisions	Org.
Roles in the organization	13	Measurement Sponsor: authorizes and supports the establishment of the measurement process	Org.
	14	Measurement process owner: responsible for the specification of what is measured, when and how, i.e., the measurement process	Org.
	15	Stakeholder: has a share, claim or interest in a system/its characteristics that meet their needs and expectations	Org.
	16	Measurement User: uses the measurement information products in his/her work	Org.

Table A.1.: Full statements of the team maturity area — Set-up [73]

A.2.2. Team maturity area — Commitment

Sub-area	No.	Statement	Level
Team's-commitment	17	The metrics team improves/evolves its ways-of-working, e.g., the team strives, continuously, to improve the time it takes to develop information products	Team
	18	The metrics team identifies new information needs that should be addressed by the organization, e.g., the team identifies and supports new projects and teams	Team
	19	The metrics team identifies new stakeholders, e.g. the team identifies and supports employees who become project and team leaders for the first time	Team
	20	The metrics team provides continuous training to the organization to disseminate the competence, e.g., the team organizes courses, presentations, and/or seminars covering all/parts of a measurement program	Team
	21	The metrics team conducts research projects, e.g., the team conducts research projects under the guidance and/or participation of academic personnel. The research projects should result in (a) the improvement of the measurement program(s) of the organization, and, (b) academic publications	Team

Organization's commitment	22	The metrics team receives management support, e.g., the line supports the team financially so that it can acquire new tools, software programs, hardware, etc., and employ new team members when needed	Org.
	23	The organization utilizes the metrics team for metrics-related issues, e.g., the metrics team is the first choice of contact when stakeholders and measurement users need new measures	Org.
	24	The metrics team receives explicit management support for research, e.g., the team receives time to conduct research projects	Org.
	25	A metrics community exists within the organization, e.g., a virtual metrics team, Community of Practice	Org.

Table A.2.: Full statements of the team maturity area — Commitment [73]

A.2.3. Team maturity area — Impact

Sub-area	No.	Statement	Level
On the organization	26	The metrics team provides deliverables, e.g., information products, measurement data, infrastructure, and a measurement experience base	Team
	27	The metrics deliverables are used in the organization, e.g., the status of information products is an input to the decision-making process	Org.
	28	The metrics deliverables are perceived as valuable, e.g. they are used as an integrated part of the organization's everyday work	Org.
	29	The metrics team impacts norms and practices in the organization, e.g., by providing valuable information products. It is easy for the organization to understand what has been measured and how well the organization performs	Org.
	30	The Measurement Experience Base is used by the metrics team and the organization, e.g., the team studies information stored in this database prior to developing new measures	Org.
	31	All (main) information needs of the organization are covered by the measurement systems provided by the metrics team, e.g., goals related to business, departments and projects/programs are formally established and followed-up	Org.
	32	All business areas are covered by the metrics team, e.g., all areas listed in the organization's balanced scorecard have information products developed by the metrics team	Org.
	33	The organization matures in measuring over time thanks to the efforts of the metrics team, e.g., they use metrics related-notions from standards and take data-based decisions	Team
	34	Information products include information quality indicators, i.e., information that shows if the products can be trusted or not	Team
Outside/ outreach	35	The metrics team's work has a significant positive impact on the customers'/organization's businesses, e.g., the team provides early warning information products that enable customers to be pro-active	Org.
	36	The metrics team actively participates in conferences, e.g., metrics, data science, machine learning and AI conferences	Team
	37	The metrics team is active in international standardization committees, e.g., ISO/IEC/IEEE 15939, ISO/IEC/IEEE (2017), and the ISO/IEC 25000 family, ISO/IEC (2016)	Team
	38	The metrics team has contact with other companies	Team
	39	The metrics team organizes and/or participates in Hackathons	Team

Table A.3.: Full statements of the team maturity area — Impact [73]

A.2.4. Team maturity area — Longevity

No.	**Statement**	**Level**
40	The metrics team covers more than one organizational unit/product/project, e.g., the support of the team stretches beyond the organization it belongs to	Team
41	The metrics team survives organizational changes, e.g., the team always has a designated place and function during/after any change in an organization's structure	Org.
42	The metrics team evolves to meet technology changes, e.g., the team is capable of supporting its organization as old products change and/or new products evolve	Team
43	The metrics team evolves its ways-of-working to meet its organization's new ways-of-working, e.g. when the organization goes from Waterfall to Agile ways-of-working	Team
44	The metrics team adjusts its ways-of-working to meet its organization's needs, e.g., the team either improves its efficiency if the organization needs more information products or starts to develop and use machine learning algorithms if the organization needs more advanced data handling	Team
45	The metrics team is perceived as an interesting/cool place to work, e.g., the perception of the organization is that the team has skilled and experienced engineers, works with the latest technology, has networks both inside and outside the company and in the academic world, and that working in the team results in career advancements	Org.
46	The metrics team's assignment is perceived to be valuable to the organization, e.g., its deliverable, such as information products, are a natural and necessary part of the organization's everyday work	Org.
47	The metrics team provides career development opportunities, e.g., working as Metrics team leader either acts as a springboard for managerial tasks or is considered meritorious when being evaluated for the position of Senior Designer	Org.
48	The metrics team provides competence development for its members, e.g., team members can take courses to become Data analysts and/or data scientists	Team

Table A.4.: Full statements of the team maturity area — Longevity [73]

A.2.5. Team maturity area — Efficiency/Ways-of-working

No.	**Statement**	**Level**
49	The metrics team has a robust, scalable, maintainable infrastructure, e.g., the architecture of the measurement infrastructure is based on measurement standards, data flow and integrity are monitored, the execution both of calculations and product information updates is efficient	Team
50	The metrics team has a short lead-time for establishing new measures and information products	Team
51	The metrics team can easily add/remove/change/evolve measures and information products, e.g., the team works according to well defined ways-of-working.	Team
52	The metrics team uses inter-operable tools (whenever possible), e.g. the team uses tools that are used and/or recognized by the organization	Team
53	Assessments are performed to ascertain the quality of measures used in the organization, e.g., the Measurement Sponsor, Measurement process owner, and quality managers perform assessments regularly to assess how well the team fulfills the measuring needs of its organization	Org.

Table A.5.: Full statements of the team maturity area — Efficiency/Ways-of-working [73]

A.2.6. Team maturity area — Team performance

Sub-area	**No.**	**Statement**	**Level**
Context	54	The metrics team shares a common understanding of its key internal and external stakeholders, i.e., customers, competitors, regulators, suppliers, the broader organization, and other internal teams	Team
	55	Metrics team members agree on the most important challenges facing the team, e.g., there is team consensus as to what the most important challenges are	Team
Mission	56	The metrics team has developed effective strategies to overcome obstacles and achieve its goals, e.g., when overwhelmed with tasks, the team knows how and what to prioritize	Team
	57	The metrics team regularly reviews progress against team goals and plans, e.g., documented goals and plans are followed up regularly at team meetings (e.g., Scrum or Kanban)	Team

Talent	58	The metrics team has the right combination of skills and experience, e.g., the team has a balanced mix of age and gender, and experienced and novice employees	Org.
	59	Metrics team members are all effective team players, e.g., each member contributes and supports others	Team
Norms	60	Metrics team meetings are effective and efficient, e.g., the team has defined different types of meetings, such as backlog follow-up, planning, retrospect, education, and an agenda exists for each type of meeting	Team
	61	Metrics team members communicate openly and directly with each other and gossiping is rare, e.g., every member feels he/she can express his/her opinion freely. Feedback from/to the team is constructive	Team
	62	Metrics team members are held accountable for their attitudes, behaviors, and deliverables	Org.
	63	The metrics team routinely reviews how to work together more effectively, e.g., team meetings are focused and short; each member is well aware of what others are working on	Team
Buy-in	64	Metrics team members understand how their actions contribute to the team's overall success, e.g., each team member can trace his/her contribution to a robust infrastructure, efficient ways-of-working, and valuable information products	Team
	65	Metrics team members faithfully adhere to team decisions and rules, e.g., they follow-up on decisions made, established plans, and set strategy	Team
Re-sources	66	The metrics team has the necessary level of political sponsorship to be successful, e.g,. all managerial levels support the work of the team and communicate this to its organization frequently	Org.
	67	The metrics team is empowered to make key decisions, e.g., the team itself decides how prioritize any backlog	Org.
	68	The metrics team proactively re-negotiates deliverables and/or backlog when faced with resource shortfalls, e.g., before vacation periods	Team
Courage	69	Metrics team members share a high degree of trust and collaboration, e.g., seniority and experience are respected, and team members work as one when emergencies occur	Team
	70	Metrics team members feel safe challenging one another, e.g., senior members respect newcomers, and old ways of working can be freely challenged	Team

	71	The metrics team has lively debates and even the most difficult issues are raised and handled respectfully, e.g., team meetings are characterized by energy, respect and the active participation of all members	Team
	72	The metrics team constructively resolves disagreements	Team

Table A.6.: Full statements of the team maturity area — Team performance [73]

A.3. Results of metrics team maturity assessment

A.3.1. Results in the area — Set-up

No.	Statement	Fully compl.	Partially compl.	Missing	Not rel.
1	The metrics team set-up is formal			SoS	ART
2	The metrics team has a defined assignment		SoS	ART	
3	The metrics team has dedicated resources		SoS, ART		
4	The metrics team has defined ways-of-working		SoS	ART	
5	The metrics team has an approved definition of its vision, mission, and strategic plan		SoS	ART	
6	There is a formal point of contact with the metrics team		SoS	ART	
7	Metrics team leader			SoS, ART	
8	Measurement designer		ART	SoS	
9	Measurement analyst		ART	SoS	
10	Database administrator		SoS, ART		
11	Data analyst			SoS, ART	
12	Data scientist			SoS, ART	
13	Measurement sponsor			SoS	ART
14	Measurement process owner		ART	SoS	
15	Stakeholder	SoS	ART		
16	Measurement user		SoS, ART		

Table A.7.: Assessment results of team maturity area - Set-up

A.3.2. Results in the area — Commitment

No.	Statement	Fully compl.	Partially compl.	Missing	Not rel.

17	The metrics team improves/evolves its ways-of-working		SoS	ART	
18	The metrics team identifies new information needs that should be addressed by the organization		SoS, ART		
19	The metrics team identifies new stakeholders		SoS, ART		
20	The metrics team provides continuous metrics training to the organization		ART	SoS	
21	The metrics team conducts research projects	ART		SoS	
22	The metrics team receives management support		ART		SoS
23	The organization utilizes the team for metrics-related issues			SoS, ART	
24	The team receives explicit management support for research			SoS	ART
25	A metrics community exists within the organization			SoS, ART	

Table A.8.: Assessment results of team maturity area - Commitment

A.3.3. Results in the area — Impact

No.	Statement	Fully compl.	Partially compl.	Missing	Not rel.
26	The metrics team provides metrics deliverables		SoS, ART		
27	The metrics deliverables are used in the organization	ART	SoS		
28	The metrics deliverables are perceived as valuable		SoS, ART		
29	The metrics impact norms and best practices in the organization		SoS, ART		
30	The Measurement Experience Base is used by the metrics team and the organization		SoS	ART	
31	All (main) information needs of the organization are covered by the measurement systems provided by the team			SoS, ART	

32	All business areas are covered by the metrics team			SoS, ART	
33	The organization matures in measuring, over time, thanks to the efforts of the metrics team		SoS	ART	
34	Information products include information quality indicators		ART	SoS	
35	The work of the metrics team has a significant impact on the customers'/organization's businesses			SoS, ART	
36	The metrics team actively participates in conferences			SoS	ART
37	The metrics team is active in international standardization committees		SoS		ART
38	The metrics team has contact with other companies		ART	SoS	
39	The metrics team organizes and/or participates in Hackathons			SoS	ART

Table A.9.: Assessment results of team maturity area - Impact

A.3.4. Results in the area — Longevity

No.	Statement	Fully compl.	Partially compl.	Missing	Not rel.
40	The metrics team covers more than one organizational unit/product/project		ART	SoS	
41	The metrics team survives organizational changes				SoS, ART
42	The metrics team evolves to meet technology changes		ART	SoS	
43	The metrics team evolves its ways-of-working to meet its organization's new ways-of-working		ART	SoS	
44	The metrics team adjusts its ways-of-working, to meet its organization's needs		SoS	ART	
45	The metrics team is perceived as an interesting/cool place to work			ART	SoS
46	The metrics team's assignment is perceived to be valuable to the organization		SoS, ART		

47	The metrics team provides career development opportunities				SoS, ART
48	The metrics team provides competence development for its members			SoS, ART	

Table A.10.: Assessment results of team maturity area - Longevity

A.3.5. Results in the area — Efficiency

No.	Statement	Fully compl.	Partially compl.	Missing	Not rel.
49	The metrics team has a robust, scalable, maintainable infrastructure		SoS	ART	
50	The metrics team has short lead-time for establishing new measures and information products				SoS, ART
51	The metrics team can easily add/remove/change/evolve measures and information products		SoS	ART	
52	The metrics team uses inter-operable tools (whenever possible)		SoS, ART		
53	Assessments are performed to ascertain the quality of measures used in the organization		SoS, ART		

Table A.11.: Assessment results of team maturity area - Efficiency/Ways-of-working

A.3.6. Results in the area — Team performance

No.	Statement	Fully compl.	Partially compl.	Missing	Not rel.
54	The metrics team shares a common understanding of its key internal and external stakeholders		SoS	ART	
55	Metrics team members agree on the most important challenges facing the team		SoS, ART		
56	The metrics team has developed effective strategies to overcome obstacles and achieve its goals		SoS, ART		

57	The metrics team regularly reviews progress against team goals and plans		SoS, ART		
58	The metrics team has the right combination of skills and experience		SoS, ART		
59	Metrics team members are all effective team players		SoS, ART		
60	Metrics team meetings are effective and efficient	ART	SoS		
61	Metrics team members communicate openly and directly with each other and gossiping is rare	ART	SoS		
62	Metrics team members are held accountable for their attitudes, behaviors, and deliverables		SoS, ART		
63	The metrics team routinely reviews how to work together more effectively	ART	SoS		
64	Metrics team members understand how their actions contribute to the team's overall success		SoS, ART		
65	Metrics team members faithfully adhere to team decisions and rules		SoS, ART		
66	The metrics team has the necessary level of political sponsorship to be successful	ART	SoS		
67	The metrics team is empowered to make key decisions	ART	SoS		
68	The metrics team proactively re-negotiates deliverables and/or backlog when faced with resource shortfalls	SoS, ART			
69	Metrics team members share a high degree of trust and collaboration	ART	SoS		
70	Metrics team members feel safe challenging each other	ART	SoS		
71	The metrics team has lively debates and even the most difficult issues are raised and handled respectfully	SoS, ART			
72	The metrics team constructively resolves disagreements	SoS, ART			

Table A.12.: Assessment results of team maturity area - Team performance

A.4. Evaluation surveys

A.4.1. General question

1. Which of the roles below applies to you?

A.4.2. Questions about the goal-based metrics introduction approach

1. The approach should engage everyone involved in planning, defining, maintaining, operating, and managing metrics, if possible, to increase awareness and adoption of metrics.

2. The approach starts with an as-is analysis of the already existing metrics and their goal coverage. In this way, potentials for the introduction of metrics based on goals can be identified.

    a) How valuable do you think the as-is analysis is as a starting point for the approach?

3. The approach envisages collecting and prioritizing the agile software development goals of all persons responsible for this. The metrics team should include the as-is analysis of the already existing metrics and their goal coverage. If there are too many goals, the approach recommends limiting the number of goals to keep the metrics candidate list manageable.

    a) How valuable do you think is the goal-based metrics adoption?

    b) How useful do you think is the prioritization and recommended limitation of goals?

    c) How helpful do you think is the inclusion of the goal coverage in prioritizing the goals?

4. The approach assigns at least one question to each goal. The questions should help to operationalize the goals and express the information interest regarding the metrics. The metrics should end up answering the question if possible. A hypothesis can also be added to each question to compare the expected outcome with the measured outcome.

    a) How useful are the questions as a link between goals and metrics?

    b) How valuable is the hypothesis for achieving a learning effect by comparing expected and measured outcome?

5. The approach looks for metric candidates that cover the respective goals and answer the questions on a quantitative level. The metrics catalog with the goal-metrics mapping and the agile frameworks with concrete proposals for metrics serve as aids. There is also the possibility to adapt the metric candidates or to define new metrics. Then one selects the desired metrics from the candidate list.

    a) How useful do you think it is to search for candidate metrics and select specific metrics?

b) How helpful is the metrics catalog in searching and selecting metrics as part of the approach?

c) How important is the flexibility to adapt and redefine metrics?

6. After selecting the metrics, the preparation of the data collection follows according to the approach. For this purpose, a measurement plan is created that defines the object, scope, time or event, data source, responsibility and the means for each metrics measurement.

   a) How important is the step of preparing the data collection?

   b) How helpful do you consider the measurement plan with its components?

7. The analysis plan next determines when, by whom, and how the metrics data will be analyzed. The values of the metrics are then interpreted in terms of answering the questions, testing the hypotheses, and achieving the goals.

   a) How important do you consider the analysis plan and the subsequent interpretation along the questions, hypotheses and goals?

8. The approach suggests a regular review of goal achievement and coverage, because goals can change and, thus, metrics need to be adjusted as well. Goal coverage can also be insufficient, for example, so that new metrics have to be added, or existing metrics no longer add value. The review should be integrated into the iterative process of the agile framework.

   a) How valuable do you think is the regular review of the goal achievement and coverage?

   b) How useful is the integration into the flow of the agile framework you are using in your company?

9. How do you evaluate the goal-based metrics adoption approach overall?

10. I am convinced that the approach will help the company to select the right metrics and to make our goals verifiable.

11. Are there steps or aspects of the approach that you would omit? If so, which ones and why?

12. Are there steps or specific aspects of the approach that you would change? If so, which ones and why?

13. Are there steps or aspects you would add to the approach? If so, which ones and why?

A.4.3. Questions about the measurement program management improvement approach

Based on an Excel template consisting of almost 70 statements, a radar chart is generated by filling it in. The radar chart evaluates the maturity of the measurement program based on six dimensions.

1. As part of the metrics team's improvement approach, the maturity model gave us an overview of the strengths and weaknesses of the current measurement program.
2. The maturity assessment is easy to perform.
3. The maturity assessment covers the relevant and essential characteristics of a successful metrics team.
4. The maturity assessment has revealed the potentials of the measurement program, so we now know what we should work on.
5. The maturity assessment helps us professionalize the measurement program and the metrics team.
6. We plan to conduct the maturity assessment at regular intervals in the future.
7. What is your overall opinion of the measurement program management improvement approach?
8. Are there steps or aspects of the approach that you would omit? If so, which ones and why?
9. Are there steps or specific aspects of the approach that you would change? If so, which ones and why?
10. Are there steps or aspects you would add to the approach? If so, which ones and why?

Bibliography

[1] P. Abrahamsson, J. Ronkainen, O. Salo, and J. Warsta. *Agile software development methods: Review and Analysis*. 2002.

[2] P. Abrahamsson, K. Conboy, and X. Wang. *'Lots done, more to do': the current state of agile systems development research*. 2009.

[3] P. Abrahamsson, N. Oza, and M. T. Siponen. "Agile software development methods: A comparative review". In: *Agile software development* (2010), pp. 31–59.

[4] M. Alqudah and R. Razali. "A review of scaling agile methods in large software development". In: *International Journal on Advanced Science, Engineering and Information Technology* 6.6 (2016), pp. 828–837.

[5] S. W. Ambler and M. Lines. *Disciplined agile delivery: A practitioner's guide to agile software delivery in the enterprise*. IBM press, 2012.

[6] H. Asuncion, D. Socha, K. Sung, S. Berfield, and W. Gregory. "Serious game development as an iterative user-centered agile software project". In: *Proceedings of the 1st International workshop on games and software engineering*. 2011, pp. 44–47.

[7] 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.

[8] 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.

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

[10] K. Beck, M. Beedle, A. Van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, et al. "Manifesto for agile software development". In: (2001).

[11] K. Beck, M. Hendrickson, and M. Fowler. *Planning extreme programming*. Addison-Wesley Professional, 2001.

[12] B. Boehm. "Get ready for agile methods, with care". In: *Computer* 35.1 (2002), pp. 64–69.

[13] M. P. Boerman, Z. Lubsen, D. A. Tamburri, and J. Visser. "Measuring and monitoring agile development status". In: *2015 IEEE/ACM 6th International Workshop on Emerging Trends in Software Metrics*. IEEE. 2015, pp. 54–62.

[14] A. Boyd. "The goals, questions, indicators, measures (GQIM) approach to the measurement of customer satisfaction with e-commerce Web sites". In: *Aslib proceedings*. MCB UP Ltd. 2002.

[15] D. Bustard, G. Wilkie, and D. Greer. “The maturation of agile software development principles and practice: Observations on successive industrial studies in 2010 and 2012”. In: *2013 20th IEEE International Conference and Workshops on Engineering of Computer Based Systems (ECBS)*. IEEE. 2013, pp. 139–146.

[16] P. R. Carlile. “A pragmatic view of knowledge and boundaries: Boundary objects in new product development”. In: *Organization science* 13.4 (2002), pp. 442–455.

[17] T.-H. 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*. IEEE. 2009, pp. 29–35.

[18] M. Choras, T. Springer, R. Kozik, L. Lopez, S. Martinez-Fernandez, P. Ram, P. Rodriguez, and X. Franch. “Measuring and improving agile processes in a small-size software development company”. In: *IEEE access* 8 (2020), pp. 78452–78466.

[19] A. Cockburn. “Agile Software Development, December 2001”. In: *paperback. Addison-Wesley Professional* (2001), p. 256.

[20] K. Conboy. “Agility from first principles: Reconstructing the concept of agility in information systems development”. In: *Information systems research* 20.3 (2009), pp. 329–354.

[21] K. Conboy and B. Fitzgerald. “Toward a conceptual framework of agile methods: a study of agility in different disciplines”. In: *Proceedings of the 2004 ACM workshop on Interdisciplinary software engineering research*. 2004, pp. 37–44.

[22] S. Coyle and J. Barata. “Socially-Constructed Metrics for Agile Quality: An Action-Research Study”. In: (2016).

[23] G. Curphy. “The Rocket Model: Practical Advice for Building High-Performing Teams”. In: (2012).

[24] C. W. Davis. *Agile Metrics in Action: How to measure and improve team performance*. Manning Publ., 2015.

[25] 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.

[26] 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.

[27] 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.

[28] T. Dingsøyr, N. B. Moe, T. E. Fægri, and E. A. Seim. “Exploring software development at the very large-scale: a revelatory case study and research agenda for agile method adaptation”. In: *Empirical Software Engineering* 23.1 (2018), pp. 490–520.

[29] T. Dingsøyr, S. Nerur, V. Balijepally, and N. B. Moe. *A decade of agile methodologies: Towards explaining agile software development*. 2012.

[30] M. Domino, A. Hevner, and R. W. Collins. "Applying agile software development processes to global virtual teams: A study of communication modalities". In: *Proceedings of the 2002 ACM SIGCPR conference on Computer personnel research*. 2002, pp. 76–78.

[31] W. W. Eckerson. *Performance dashboards: measuring, monitoring, and managing your business*. John Wiley & Sons, 2010.

[32] H. Edison, X. Wang, and K. Conboy. "Comparing Methods for Large-Scale Agile Software Development: A Systematic Literature Review". In: *IEEE Transactions on Software Engineering* (2021).

[33] J. A. Espinosa, S. A. Slaughter, R. E. Kraut, and J. D. Herbsleb. "Familiarity, complexity, and team performance in geographically distributed software development". In: *Organization science* 18.4 (2007), pp. 613–630.

[34] 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.

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

[36] T. L. Ferris. "Managing Virtual Teams: Getting the Most From Wikis, Blogs, and Other Collaborative Tools [Book review; Brown, MK et al.; 2007]". In: *IEEE transactions on professional communication* 50.3 (2007), pp. 263–265.

[37] M. Fowler, J. Highsmith, et al. "The agile manifesto". In: *Software Development* 9.8 (2001), pp. 28–35.

[38] A. Gopal, M. S. Krishnan, T. Mukhopadhyay, and D. R. Goldenson. "Measurement programs in software development: determinants of success". In: *IEEE Transactions on software engineering* 28.9 (2002), pp. 863–875.

[39] 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.

[40] G. Gruver and T. Mouser. *Leading the transformation: Applying agile and devops principles at scale*. IT Revolution, 2015.

[41] J. R. Hackman and R. J. Hackman. *Leading teams: Setting the stage for great performances*. Harvard Business Press, 2002.

[42] T. Hall and N. Fenton. "Implementing effective software metrics programs". In: *IEEE software* 14.2 (1997), pp. 55–65.

[43] G. K. Hanssen and T. E. Fægri. "Process fusion: An industrial case study on agile software product line engineering". In: *Journal of Systems and Software* 81.6 (2008), pp. 843–854.

[44] A. R. Hevner, S. T. March, J. Park, and S. Ram. "Design science in information systems research". In: *MIS quarterly* (2004), pp. 75–105.

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

[46] H. Holmström, B. Fitzgerald, P. J. Ågerfalk, E. Ó. Conchúir, et al. "Agile practices reduce distance in global software development". In: *Information systems management* 23.3 (2006), pp. 7–18.

[47] B. Horlach, I. Schirmer, and P. Drews. "Agile portfolio management: design goals and principles". In: (2019).

[48] M. Hummel. "State-of-the-art: A systematic literature review on agile information systems development". In: *2014 47th Hawaii International Conference on System Sciences*. IEEE. 2014, pp. 4712–4721.

[49] L. Jiang and A. Eberlein. "An analysis of the history of classical software development and agile development". In: *2009 IEEE International Conference on Systems, Man and Cybernetics*. IEEE. 2009, pp. 3733–3738.

[50] T. Jokela and P. Abrahamsson. "Usability assessment of an extreme programming project: Close co-operation with the customer does not equal to good usability". In: *International Conference on Product Focused Software Process Improvement*. Springer. 2004, pp. 393–407.

[51] C. Jones. *Applied software measurement*. McGraw-Hill Education, 2008.

[52] R. S. Kaplan, D. P. Norton, et al. "The balanced scorecard: measures that drive performance". In: *Harvard business review* 83.7 (2005), p. 172.

[53] H. Kerzner. *Project management metrics, KPIs, and dashboards: a guide to measuring and monitoring project performance*. John Wiley & Sons, 2017.

[54] P. Kettunen et al. "Agile software development in large-scale new product development organization: team level perspective". In: (2009).

[55] P. Kettunen, M. Laanti, F. Fagerholm, and T. Mikkonen. "Agile in the era of digitalization: a finnish survey study". In: *International Conference on Product-Focused Software Process Improvement*. Springer. 2019, pp. 383–398.

[56] M. Khurum, T. Gorschek, and M. Wilson. "The software value map—an exhaustive collection of value aspects for the development of software intensive products". In: *Journal of software: Evolution and Process* 25.7 (2013), pp. 711–741.

[57] T. Kilpi. "Implementing a software metrics program at Nokia". In: *IEEE software* 18.6 (2001), pp. 72–77.

[58] I. Korpivaara, T. Tuunanen, and V. Seppänen. "Performance Measurement in Scaled Agile Organizations". In: *Proceedings of the Annual Hawaii International Conference on System Sciences*. University of Hawai'i at Manoa. 2021.

[59] H. Koziolek. "Goal, question, metric". In: *Dependability metrics*. Springer, 2008, pp. 39–42.

[60] J. Krebs. *Agile portfolio management*. Microsoft Press, 2008.

[61] T. C. Krell. "Organizational longevity and technological change". In: *Journal of Organizational Change Management* (2000).

[62] 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.

[63] M. Laanti, J. Similä, and P. Abrahamsson. "Definitions of agile software development and agility". In: *European Conference on Software Process Improvement*. Springer. 2013, pp. 247–258.

[64] L. Lagerberg, T. Skude, P. Emanuelsson, K. Sandahl, and D. Ståhl. "The impact of agile principles and practices on largescale software development projects". In: *Linkopings university, SE-581* 83 (2013).

[65] M. Laitinen, M. E. Fayad, and R. P. Ward. "Thinking objectively: The problem with scalability". In: *Communications of the ACM* 43.9 (2000), pp. 105–107.

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

[67] T.-P. Liang, C.-C. Liu, T.-M. Lin, and B. Lin. "Effect of team diversity on software project performance". In: *Industrial Management & Data Systems* (2007).

[68] M. Lindvall, V. Basili, B. Boehm, P. Costa, K. Dangle, F. Shull, R. Tesoriero, L. Williams, and M. Zelkowitz. "Empirical findings in agile methods". In: *Conference on extreme programming and agile methods*. Springer. 2002, pp. 197–207.

[69] B. List, R. M. Bruckner, and J. Kapaun. "Holistic software process performance measurement from the stakeholders' perspective". In: *16th International Workshop on Database and Expert Systems Applications (DEXA'05)*. IEEE. 2005, pp. 941–947.

[70] V. Mahnic and N. Zabkar. "Using COBIT indicators for measuring scrum-based software development". In: *Wseas transactions on computers* 7.10 (2008), pp. 1605–1617.

[71] S. Martinez-Fernandez, A. Jedlitschka, L. Guzmán, and A. M. Vollmer. "A quality model for actionable analytics in rapid software development". In: *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE. 2018, pp. 370–377.

[72] J. McGarry. *Practical software measurement: objective information for decision makers*. Addison-Wesley, Boston, MA, 2001.

[73] W. Meding, M. Staron, and O. Söder. "MeTeaM—A method for characterizing mature software metrics teams". In: *Journal of Systems and Software* (2021), p. 111006.

[74] M. B. Miles, A. M. Huberman, and J. Saldaña. *Qualitative data analysis: A methods sourcebook*. Sage publications, 2014.

[75] D. Millington and J. Stapleton. "Developing a RAD standard". In: *IEEE Software* 12.5 (1995), pp. 54–55.

[76] E. Miranda and P. Bourque. "Agile monitoring using the line of balance". In: *Journal of Systems and Software* 83.7 (2010), pp. 1205–1215.

[77] S. Misra, V. Kumar, U. Kumar, K. Fantazy, and M. Akhter. "Agile software development practices: evolution, principles, and criticisms". In: *International Journal of Quality & Reliability Management* (2012).

[78] K. Mohan, B. Ramesh, and V. Sugumaran. "Integrating software product line engineering and agile development". In: *IEEE software* 27.3 (2010), pp. 48–55.

[79] 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.

[80] S. Nerur and V. Balijepally. "Theoretical reflections on agile development methodologies". In: *Communications of the ACM* 50.3 (2007), pp. 79–83.

[81] R. L. Nord and J. E. Tomayko. "Software architecture-centric methods and agile development". In: *IEEE software* 23.2 (2006), pp. 47–53.

[82] R. J. Offen and R. Jeffery. "Establishing software measurement programs". In: *IEEE Software* 14.2 (1997), pp. 45–53.

[83] N. Oza and M. Korkala. "Lessons Learned In Implementing Agile Software Development Metrics." In: *UKAIS*. 2012, p. 38.

[84] F. Paetsch, A. Eberlein, and F. Maurer. "Requirements engineering and agile software development". In: *WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003.* IEEE. 2003, pp. 308–313.

[85] D. Parmenter. "Key Performance Indicators: Developing, Implementing, and Using Winning KPIs, John Whiley & Sons". In: *Inc, Hoboken, New Yersey* (2007).

[86] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee. "A design science research methodology for information systems research". In: *Journal of management information systems* 24.3 (2007), pp. 45–77.

[87] M. Pikkarainen and A. Mantyniemi. "An approach for using CMMI in agile software development assessments: experiences from three case studies". In: (2006).

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

[89] 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.

[90] 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. 2018, pp. 272–287.

[91] P. Ram, P. Rodriguez, M. Oivo, and S. Martinez-Fernandez. "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. 2019, pp. 14–23.

[92] D. J. Reifer, F. Maurer, and H. Erdogmus. "Scaling agile methods". In: *IEEE software* 20.4 (2003), pp. 12–14.

[93] K. Rolland, T. Dingsoyr, B. Fitzgerald, and K.-J. Stol. "Problematizing agile in the large: alternative assumptions for large-scale agile development". In: *39th International Conference on Information Systems*. Association for Information Systems (AIS). 2016, pp. 1–21.

[94] D. Rombach. *Aligning Organizations Through Measurement The GQM+ Strategies Approach*. Springer, 2014.

[95] P. Runeson and M. Höst. "Guidelines for conducting and reporting case study research in software engineering". In: *Empirical software engineering* 14.2 (2009), pp. 131–164.

[96] H.-P. Samios. "Overcoming Traditional Project Release Reporting with an Agile Approach Focused on Change". In: *2012 Agile Conference*. IEEE. 2012, pp. 131–135.

[97] A. Sandberg, L. Pareto, and T. Arts. "Agile collaborative research: Action principles for industry-academia collaboration". In: *IEEE software* 28.4 (2011), pp. 74–83.

[98] P. Schuh. *Integrating agile development in the real world*. Charles River Media, Inc., 2004.

[99] K. Schwaber. "Scrum development process. Business object design and implementation". In: *10th Annual ACM Conference on Object Oriented Programming Systems, Languages, and Applications Addendum to the Proceedings. ACM/SIGPLAN October*. 1995, pp. 117–134.

[100] K. Schwaber and J. Sutherland. URL: https://www.scrum.org/resources/what-is-scrum.

[101] K. Schwaber and J. Sutherland. *The Scrum Guide.* Nov. 2020. URL: https://scrumguides.org/.

[102] M. K. Sein, O. Henfridsson, S. Purao, M. Rossi, and R. Lindgren. "Action design research". In: *MIS quarterly* (2011), pp. 37–56.

[103] D. R. van Solingen and E. W. Berghout. *The Goal/Question/Metric Method: a practical guide for quality improvement of software development*. McGraw-Hill, 1999.

[104] M. Staron and W. Medig. "Factors determining long-term success of a measurement program: an industrial case study". In: *e-Informatica Software Engineering Journal* 5.1 (2011).

[105] M. Staron and W. Meding. "Software Development Measurement Programs". In: *Springer. https://doi. org/10.1007/978-3-319-91836-5* 10 (2018), p. 3281333.

[106] *State of Agile Survey*. URL: https://stateofagile.com/#ufh-i-615706098-14th-annual-state-of-agile-report/7027494.

[107] 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.

[108] 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. 2018, pp. 199–215.

[109] V. Subramaniam and A. Hunt. *Practices of an agile developer: Working in the real world*. Pragmatic Bookshelf, 2006.

[110] J. Sutherland and C. Scrum Inc. *Read the Scrum@Scale Guide Today*. Feb. 2021. URL: `https://www.scrumatscale.com/scrum-at-scale-guide/`.

[111] F. M. Tobisch. "Design of a Metric Catalog for Large-Scale Agile Software Development". 2021.

[112] N. Treiber. "Goals and KPIs in Large-scale Agile Development: A Systematic Literature Review". 2021.

[113] R. Turner and A. Jain. "Agile meets CMMI: Culture clash or common cause?" In: *Conference on Extreme Programming and Agile Methods*. Springer. 2002, pp. 153–165.

[114] X. Wang, K. Conboy, and O. Cawley. ""Leagile" software development: An experience report analysis of the application of lean approaches in agile software development". In: *Journal of Systems and Software* 85.6 (2012), pp. 1287–1299.

[115] L. Williams. "What agile teams think of agile principles". In: *Communications of the ACM* 55.4 (2012), pp. 71–76.

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

[117] R. K. Yin. *Case study research and applications: design and methods*. Sixth. SAGE, 2018.

[118] A. Zaitsev, U. Gal, and B. Tan. "Reviewing the role of the Agile Manifesto and agile methods in literature". In: (2018).