DEPARTMENT OF INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Master's Thesis in Information Systems

Establishing and Reporting Goals in Large-Scale Agile Software Development

Moritz Schüll

DEPARTMENT OF INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Master's Thesis in Information Systems

Establishing and Reporting Goals in Large-Scale Agile Software Development

Festlegen und Berichten von Zielen in der skalierten agilen Softwareentwicklung

Author:	Moritz Schüll
Supervisor:	Prof. Dr. Florian Matthes
Advisor:	Pascal Philipp, M. Sc.
Submission Date:	September 15, 2021

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

Munich, September 15, 2021 Moritz Schüll

Abstract

Agile software development methodologies are becoming increasingly interesting for application in large-scale projects. While agile methodologies have been originally designed for usage by small, colocated teams to develop software products, their benefits make them attractive for larger use cases as well. Their frequent inspect and adapt cycles and focus on close customer collaboration allow for better adjustment to changing environments and requirements. They promise results that satisfy customer needs better than those of traditional methodologies. However, applying agile methodologies in large-scale settings comes with additional challenges. Working with multiple teams or on multiple products at the same time yields higher coordination and communication efforts. Dependencies may constrain the autonomy of individual agile teams, a central aspect common to agile methodologies. Thus, it is important to establish shared goals for collaborating agile teams, while still allowing for autonomy of individual teams. With the introduction of agile methodologies at scale also comes the need to be able to measure performance not only of individual teams but also on higher aggregation of products and portfolios. Due to faster iterations and production of intermediate work results, agile methodologies are challenging the existing reporting methodologies in large organizations. Yet, reporting and goal-setting in large-scale agile software development are not extensively covered in extant literature.

Given these considerations and research gap, this master's thesis investigates how goal-setting and reporting in large-scale agile software development is currently done in practice. By collaborating with a large German car manufacturer and conducting 23 interviews with 17 practitioners of agile methodologies in large-scale environments, the state-of-the-art is investigated and potential areas for improvement and recommendations are derived. Challenges with existing approaches are documented. Finally, a process model that comprises the key activities for goal-setting and reporting is developed. Based on the identified challenges potential solutions are proposed. The created statements and models are evaluated positively in the case organization, with few adjustments made after the evaluation.

In summary, this thesis contributes to the scientific conversation on goal-setting and reporting in large-scale agile software development by documenting approaches applied in practice, the reasons behind their usage, and current challenges. The overall findings are consolidated in a process model of goal-setting and reporting in large-scale agile development. It also provides suggestions and guidance for practitioners.

Contents

Outline of the Thesis

CHAPTER 1: INTRODUCTION

The introduction describes the research objectives of this thesis. It outlines the chosen research approach to answer the research questions and achieve these objectives. The chapter explains the motivation and describes why research in this area is relevant to the current scientific discussion.

CHAPTER 2: FOUNDATIONS

The second chapter defines relevant concepts and reviews existing literature. It lays out foundations on large-scale agile software development, goal-setting, and reporting.

CHAPTER 3: RELATED WORK

The third chapter summarizes and evaluates existing research from the area of large-scale agile software development, with focus on goal-setting and reporting. Results of extant research are reviewed and the research of this thesis is demarcated.

CHAPTER 4: CASE STUDY

The fourth chapter presents the case study of this thesis. The methodology is described, the case organization is presented, and the results are analyzed. The results are practices, metrics, challenges, and reasons. They answer the first two research questions.

CHAPTER 5: THEORETICAL MODEL

The fifth chapter develops two artifacts to answer the third research question. Propositions are derived to address the identified challenges. Using these propositions, the chapter describes a generalized process model. The model comprises key activities for goal-setting and reporting in large-scale agile software development.

CHAPTER 6: ARTIFACT EVALUATION AND IMPROVEMENT

The sixth chapter describes how the artifacts developed in chapter five are evaluated. The chapter presents the used methodology, followed by the results of the evaluation. Opportunities for improvement of the artifacts are described, based on the evaluation.

CHAPTER 7: DISCUSSION

The discussion collects the key findings identified in this project. Potential limitations of the presented research are discussed.

CHAPTER 8: CONCLUSION

The final chapter summarizes the work conducted in this research project and outlines possibilities for future work.

1. Introduction

The first chapter presents the motivation, research objectives, and approach of this master's thesis. Section 1.1 describes the motivation behind the research project, and why it is interesting and relevant to pursue. Section 1.2 explains the objectives and research questions that we set out to answer in this thesis. Finally, Section 1.3 outlines the scientific approach employed to achieve the stated objectives.

1.1. Motivation

In today's organizations, diverse project management methodologies are used for software development [65]. While all of these methodologies aim at developing software, they differ significantly in how they view the development process. While traditional methodologies consider the software development process to be of defined nature, it is a commonality of all agile methodologies to consider software development as an empirical process [66]. Empirical processes require frequent feedback and adjustment to changing environments to allow for adequate reaction to unpredictable demands and requirements [66]. This common key characteristic of agile methodologies makes them suitable for projects with unforeseeable or changing demands, which is the case for many software development projects [66].

As of today, agile methodologies have become the dominant type of software development methodologies [26]. In the 2020 issue of the yearly State of Agile report conducted by VersionOne Inc. 95% of the participating organizations indicated to be using agile methodologies [26]. Likewise, the research area around large-scale agile development has seen increasing activity, with 47% of publications in this area being from the past two years [63]. Considering the formal inception of agile methodologies in 2001, over the past 20 years they have gained significant traction. In 2001, several authors compiled the central values and principles of their software development methodologies and coined the term "agile" [6]. These agile methodologies include, among others, the SCRUM framework by Schwaber and Sutherland [54] and eXtreme Programming (XP) by Beck [5]. While such methodologies had been used before 2001, the Agile Manifesto sparked a conversation around the agile approach [66].

Agile methodologies had initially been designed for usage in non-critical projects developed by small, colocated teams [14, 66]. Nevertheless, given their proven benefits, agile methodologies have become increasingly appealing to larger organizations as well [14]. As a consequence, several scaling agile frameworks have emerged that focus on the appli-

cation of agile methodologies in large-scale environments with multiple products and development teams. The Scaling Agile Frameworks (SAFe) and Scrum of Scrums are among the most popular scaling agile frameworks, as of today [26]. However, applying agile methodologies at scale comes with several new challenges, such as additional coordination overhead, dependencies between teams, and more [14]. Identifying these challenges as well as potential solutions to them is the subject of an ongoing research conversation. At the International Conference on Agile Software Development 2014 (XP2014 Conference), Dingsøyr and Moe [16] organized a workshop to lay out a research agenda for challenges and solutions in large-scale agile software development. In the resulting research agenda, among others, the participants identified *Organization of large development efforts*[1], *Key performance indicators in large development efforts*[2], and *Knowledge sharing and improvement*[3] as top priority research areas [16]. In a more recent installment of their workshop at XP2019 Conference, Moe et al. [43] identify lack of clear and common goals as a top barrier to autonomy of agile teams, and put measurement of performance on the research agenda [43]. Further, in a recent mapping study on the state-of-the-art of large-scale agile software development, Uludağ et al. find that there is no notable research stream on performance measurement of large-scale agile initiatives [63]. This is where this master's thesis seeks to contribute to the current research conversation. On the one hand, this thesis investigates how goals are established in large-scale agile organizations, contributing to the organizational models and project management area of the research agenda. On the other hand, by also investigating reporting processes and approaches in large-scale agile organizations, a contribution to the key performance indicators and knowledge sharing and improvement areas of the research agenda shall be made.

1.2. Research Objectives

Following the previously outlined motivation, we formulate three research questions (RQs) that we seek to answer with this research project. The structure of this master's thesis is oriented around these RQs.

> **Research Question 1:** How are goals in large-scale agile software development established and reported at the case organization?

The first research question seeks to describe the current situation of goal-setting and reporting in large-scale agile software development in practice. To answer this research question we conduct a case study at a large German car manufacturer. The question investigates which approaches practitioners at the case organization are using at the moment.

[1]Description: "Organizational models, portfolio management, project management, agile product-line engineering." [16]
[2]Description: "Identify appropriate metrics are [*sic*] to monitor progress and support transparency." [16]
[3]Description: "How to ensure feedback for learning, use of knowledge networks and learning practices." [16]

> **Research Question 2:** What are challenges and reasons for establishing and reporting goals in large-scale agile software development?

The second research question is concerned with challenges that practitioners are facing with their current approaches of goal-setting and reporting. Further, the question also is about the reasons behind why the current goals and approaches are used. It seeks to understand what the motivation of practitioners was to implement goal-setting and reporting as they did. To answer this research question we also employ the case study used to answer RQ1.

> **Research Question 3:** How can these challenges for establishing and reporting goals in large-scale agile software development be addressed?

The third research question seeks to investigate how the identified challenges from RQ2 can be addressed. To answer this research question, a list of propositions is formulated that we hypothesize may address the identified challenges. These propositions are presented to and evaluated by practitioners. Based on these propositions a process model for goal-setting and reporting in large-scale agile development is built.

1.3. Approach

To answer the research questions, this thesis follows a Design Science Research (DSR) approach [23, 47]. The goal of Design Science Research is to create innovative artifacts that solve problems in organizations or enhance organizational capabilities [23]. Specifically, we follow the principles of Action Design Research (ADR), a DSR methodology designed for "generating prescriptive design knowledge through building and evaluating ensemble artifacts in an organizational setting" [57]. Compared to plain DSR, ADR emphasizes that artifact building and evaluation are not two separate, sequential steps [57]. Instead ADR integrates artifact building and evaluation in an iterative way, with formative evaluations that "contribut[e] to the refinement of the artifact" [57].

The ADR methodology defines four stages [57]. In the Problem Formulation stage a research opportunity is identified based on existing research or practical challenges in organizations [57]. The research objective is formulated based on the identified problems. In this thesis, Chapter 4 represents the problem formulation stage. Anticipated from a lack of existing research, this chapter conducts a case study to investigate existing approaches and challenges of goal-setting and reporting in large-scale agile software development. The identified challenges form the problem formulation for this thesis, which will be addressed with the artifact. The case study follows the guidelines by Runeson and Höst [52], which is explained in Section 4.1. It can be classified as a single-case, embedded case study [68]. How the case study fits into the overall ADR research approach is also visualized in Figure 1.1.

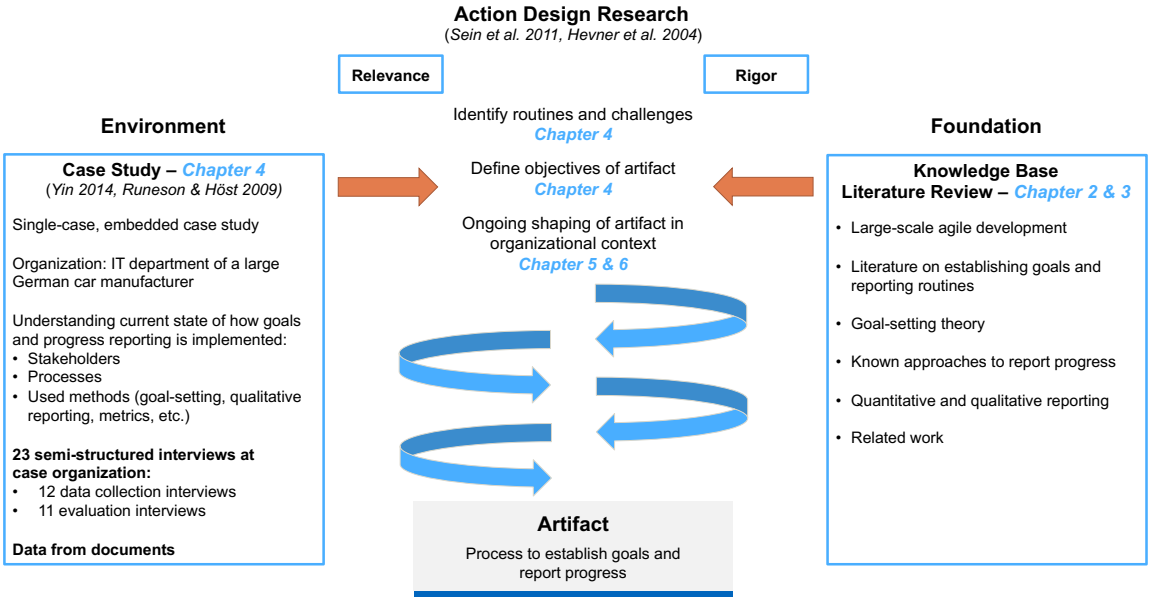


Figure 1.1.: Visualization of the overall research approach, including the Action Design Research by Sein et al. [57] and the case study research approach by Runeson and Höst [52]

The second stage of ADR is Building, Intervention, and Evaluation [57]. This stage is realized in Chapters 5 and 6 of this thesis. In Chapter 5, a theoretical model of goal-setting and reporting in large-scale agile development is built based on the initial case study, using propositions to address the identified challenges. This theoretical model, specifically the propositions that form its basis, is subsequently evaluated in Chapter 6. The artifact creation follows the methodology for theory building in software engineering by Sjøberg et al. [58], which is explained in Section 5.1. Based on the evaluation results and collected feedback, potential improvements to the propositions are discussed in the second part of Chapter 6. This realizes the third stage of ARD, Reflection and Learning [57]. The learnings and knowledge generated throughout this thesis are formalized in the process models for goal-setting and reporting, which are created based on the propositions. This can be mapped to the Formalization stage of ADR [57].

The chapters of this thesis and how they map to the stages of Action Design Research are visualized in Figure 1.2. Overall, in this thesis we conduct one full cycle of the ADR stages, followed by an outlook of how the artifacts might be improved in a potential next design iteration. The research can be classified as exploratory and improving, as defined by Runeson and Höst [52]. Exploratory, because with the case study we seek to find out how goal-setting and reporting are happening in practice. Improving, because by formulating the propositions we try to address existing challenges in practice, and to describe how a general process for goal-setting and reporting can look like.

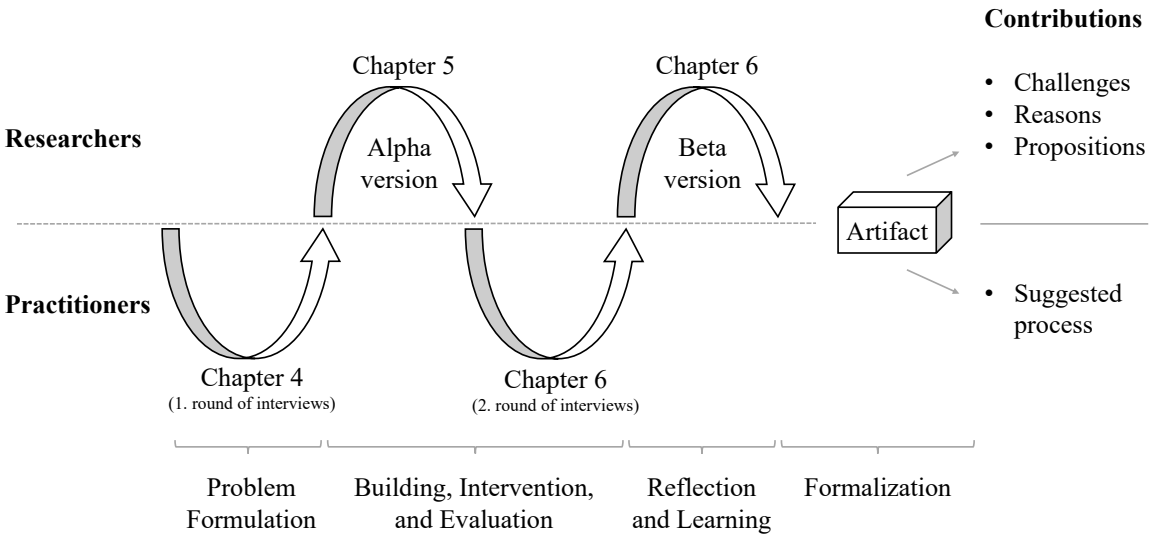


Figure 1.2.: Visualization of how the steps of Action Design Research by Sein et al. [57] map to the chapters of this thesis

A balance between relevance and rigor is central to the Design Science Research approach [23]. By reviewing and incorporating extant literature we ensure that our research is rigorous. Relevancy of our research is ensured by using insights and challenges from the case study with a real, productive organization for the problem formulation.

The remainder of this master's thesis is structured as follows. In Chapter 2 scientific foundations, on which this thesis is built on, are reviewed and explained. Chapter 3 discusses related work in the area of goal-setting and reporting in large-scale agile software development. The case study that we conducted at a German car manufacturer is presented in Chapter 4. Chapter 5 develops the artifacts, which are the propositions to address the identified challenges, and the process models for goal-setting and reporting built on these propositions. The propositions are evaluated and potential areas of improvement of the artifacts are discussed. This is presented in Chapter 6. Lastly, Chapter 7 summarizes the key findings and limitations of this thesis. Chapter 8 gives an outlook for potential future research.

2. Foundations

This chapter presents the foundations and concepts on which this thesis builds. Important concepts used in the following chapters are explained and relevant literature is presented. First, in Section 2.1 agile software development and its central values are described. This section also explains how large-scale agile software development is defined. Second, Section 2.2 explains the concepts of traditional goal-setting theory, and how this theory can be transferred into agile settings. Finally, in Section 2.3 important types of reporting and necessary concepts to understand their differences are discussed.

2.1. Large-Scale Agile Software Development

This section provides theoretical background on agile software development and defines what is considered large-scale agile in the context of this thesis. Also, two common scaling agile frameworks are described in detail.

2.1.1. Agile Software Development Methodologies

Today, agile software development methodologies are the dominant approach of developing software [26]. Several frameworks are applied by organizations to implement agility, with Scrum, Kanban, and hybrids of these two being the most popular ones [26]. The term "agile" has been coined by the Agile Manifesto, a summary of the values and principles that is common to all the agile development approaches [6, 66]. Those central values include the focus on the individuals and their interactions over formal processes or fixed tools [6]. Documentation is subordinated to working software, as is fixed contracts to frequent customer collaboration [6]. Further, following a fixed plan is disregarded in favor of the ability to adapt to changing environments and requirements [6]. In summary, while traditional development methodologies have approached software development as a defined process, agile methodologies approach it as an empirical process and incorporate frequent adjustment to changes and unforeseen circumstances [66]. As indicated by the values and principles of the Agile Manifesto, agile approaches typically do not offer a lot of guidance in evaluating and reporting progress. The official Scrum guide, for example, does not mention any metrics [56], and Ken Schwaber, the original author of Scrum, only mentions the Velocity as a metric in his description of the Scrum development process [54]. The Agile Manifesto states that "working software is the primary measure of progress" [6] when using agile methodologies.

2.1.2. Large-Scale Agile Software Development

Originally, agile software development methodologies have been designed for use in small, co-located teams [9]. However, in recent time they have also been increasingly applied to larger projects and by larger organizations [26]. Further, the research stream concerned with large-scale agile software development has seen increased activity in the past few years, with over 47% of research activity happening in the previous two years [63]. However, research has also investigated and documented several challenges that arise when applying agile methodologies to large-scale projects [14, 61]. Given this popularity and challenges, several scaling agile frameworks have emerged that try to make application of agile methodologies to scaled projects easier. Two of these frameworks, SAFe and LeSS, are explained in the following sections.

Further, for the context of this thesis, a clear definition of the term *large-scale agile* is necessary. In literature, no clear definition has emerged yet [15]. Dingsøyr et al. [15] propose a taxonomy of scale that defines various manifestations of scale. They argue against using size measures such as cost or code size as indication of the scaling size due to industry specific differences. Instead, they propose using the *coordination overhead* as an indicator of scaling size [15]. The coordination overhead in their taxonomy is based on a *7 (+/- 2)* rule of thumb [15]. Based on coordination overhead they define *small-scale* as a single team, and *large-scale* as ranging from two to nine teams because additional coordination is needed compared to the small-scale case [15]. Finally, Dingsøyr et al. [15] also define the *very large-scale* which covers cases of ten or more teams. Other authors are defining *large-scale agile* quite differently. Dingsøyr and Moe [16] give an overview of different definitions for *large-scale* that they collected at the XP2014 conference, of which many rely on size measures.

In line with this taxonomy by Dingsøyr et al. [15], this thesis defines large-scale agile development as cases where multiple teams are working together on one product using agile software development methodologies. Further, we also consider cases as large-scale agile where agile methodologies have been rolled out throughout the whole organization, thus also covering the *very large-scale* definition of Dingsøyr et al. [15].

2.1.3. Scaled Agile Framework (SAFe)

The Scaled Agile Framework (SAFe) was originally designed by Dean Leffingwell [50] in his book on "scaling software agility" [36]. Today, SAFe is the most commonly used scaling agile framework according to the 14th State of Agile Report by VersionOne Inc. [26]. SAFe is also being used by several of the interviewees of the case study that is part of this thesis. Hence, the SAFe framework is explained in this section.

SAFe builds on the values and principles of agile and lean product development [35]. It offers four different configurations, called Essential, Large Solution, Portfolio, and Full SAFe. The Full SAFe is displayed in Figure 2.1. Essential SAFe comprises the Team and the Program level that together form an Agile Release Train (ART) [35]. In the Essential

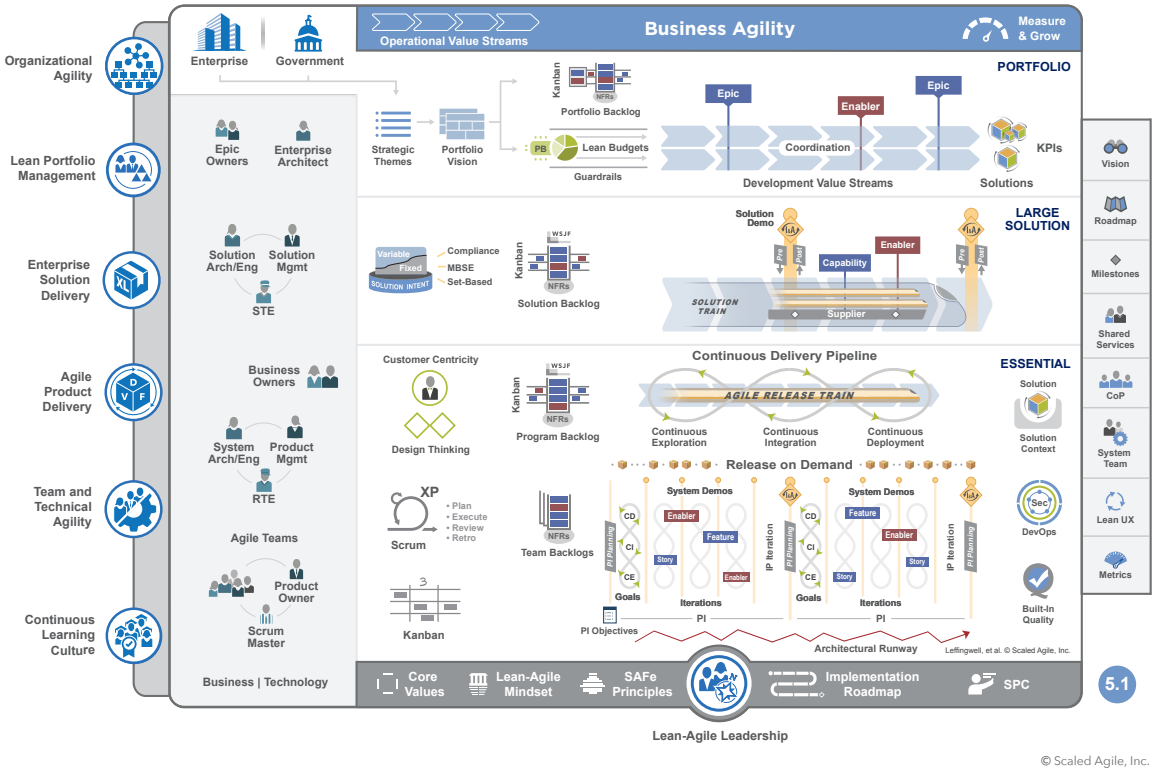


Figure 2.1.: Overview of the Full SAFe configuration, taken form the SAFe 5 guide [25]

configuration Agile Teams, Systems Architects, Product Management, Release Train Engineers (RTEs), Business Owners, and Customers are present [35]. The ART aligns all the stakeholders to a common roadmap and operates in Program Increments (PIs) [35]. Inside of the ART, the iterations of the Agile Teams are synchronized, which is called the *cadence* of the ART. A PI is "a time-boxed increment[...] for planning, execution, and inspecting and adapting" [35]. Typically, a PI is prefaced by a PI Planning and a new solution is presented at a System Demo and released at the end of a PI.

The next bigger scaling configuration in SAFe is called Large Solution. It builds on Essential SAFe and allows to coordinate the work of multiple ARTs [35]. This coordination is handled by the Solution Train [35]. The Solution Train maintains its own Solution Kanban and operates in the Solution Context, which describes the environment that the Solution will operate in [35]. Compared to Essential SAFe, the Large Solution configuration adds the roles of the Solution Train Engineer (STE), Solution Management, and Solution Architect [35]. These roles are responsible for coordination of working methodology, solution content, and solution architecture across all the involved ARTs, respectively [35]. Further, compared to Essential SAFe, Large Solution SAFe adds Pre- and Post-PI Planning events and an overall Solution Demo where individual ARTs' results are evaluated together [35].

The Portfolio configuration of SAFe is intended to make sure that the SAFe program is aligned with the overall enterprise strategy [35]. Just like Large Solution SAFe, Portfolio SAFe builds on the Essential configuration and adds several new aspects. The Value Stream is added to make sure that the used resources of the enterprise are actually providing value to the customer or the business, thus ensuring a flow of value [35]. Further, Portfolio SAFe adds a Portfolio Kanban, to make the work of the Portfolio visible, and budgeting practices that build on the Lean values [35]. The Portfolio Kanban contains Portfolio Epics which are coordinated by the Epic Owners, a new role added by the Portfolio configuration [35]. Further, Portfolio SAFe adds the roles of the Enterprise Architect and Lean Portfolio Management (LPM), which are ensuring strategic direction from a technical and investment perspective [35].

Finally, Full SAFe is the biggest configuration and comprises all the aforementioned configurations [35]. The Full SAFe configuration is displayed in Figure 2.1. It includes the Team, Program, Large Solution, and Portfolio levels.

Throughout all levels SAFe emphasizes Inspect and Adapt, as is also part of the general agile values. ARTs and value streams are conducting Inspect and Adapt workshops during which they define improvement Backlog Items based on the current state of work [35].

Goals in SAFe

Among many practices and guidelines, SAFe also provides goal-setting mechanisms and metrics as part of the framework [25]. These elements of the framework are of particular relevance since this thesis is concerned with goals and reporting.

Figure 2.2 gives an overview of the hierarchy of goals as defined by SAFe. "**Program Increment (PI) Objectives** are a summary of the business and technical goals that an Agile Team or train intends to achieve in the upcoming Program Increment (PI)" [25]. They are created by Agile Teams during the PI Planning event and represent things they want to accomplish in the next PI. PI Objectives can, but do not have to, relate to features [25]. SAFe recommends PI Objectives be formulated using the SMART technique (Specific, Measurable, Achievable, Realistic, Time-bound) [25]. During PI Planning the PI Objectives are assigned a relative business value on a scale from 1 to 10 by Business Owners [25]. In general, PI objectives are defined and rolled up bottom-up [25]. All the team

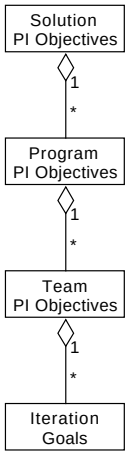


Figure 2.2.: The layers of PI Objective roll-up, based on the SAFe 5 guide [25]

objectives, once committed, are summarized into **Program PI Objectives** by the Release Train Engineer (RTE) [25]. Further, after all Agile Release Trains (ARTs) have planned their Program PI Objectives, the Solution Train Engineer summarizes them into **Solution PI Objectives** [25]. The Solution PI Objectives are used to communicate to stakeholders what the Solution Train will deliver in the PI [25].

"**Iteration Goals** are a high-level summary of the business and technical goals that the Agile Team agrees to accomplish in an Iteration" [25]. The Iteration Goals make sure teams are working towards the PI Objectives [25]. They are intended for the team to keep a larger view of what is planned to be achieved in the iteration, and to be able to understand and communicate the business value that will be delivered [25]. Further, they allow for more detailed dependency handling on the team-level [25]. The Iteration Goals are a set of statements that summarize the stories that have been planned to be in the *Iteration Backlog* [25]. The Iteration Goals are derived from the PI Objectives for the current Program Increment as well as the planned stories in the Iteration Backlog [25]. The Iteration Backlog, in turn, is influenced by the available capacity of the team for the upcoming iteration [25]. At the end of the Iteration Planning everyone commits to the Iteration Goals [25].

Reporting and Metrics in SAFe

Even though SAFe declares the actual software to be the primary measure of progress towards the defined objectives, the framework also provides several metrics. In SAFe, metrics are defined as follows:

> *Metrics are agreed-upon measures used to evaluate how well the organization is progressing toward the Portfolio, Large Solution, Program, and Team's business and technical objectives.*
>
> — Definition of metrics by the SAFe framework [35]

The metrics in SAFe are structured according to the different configurations. Table 2.1 shows an overview of several of the metrics and measurements that are part of the SAFe framework. We encourage the reader to also refer to the official SAFe website[1], as there are several visualizations provided to further explain the respective metrics and measurements.

[1] https://www.scaledagileframework.com/metrics

What is evaluated	Measurements / metrics
Agility	Business Agility Self-Assessment Organizational Agility Self-Assessment
Internal and external progress	Lean Portfolio Metrics
Lean and Systems Thinking	Lean Portfolio Management Self-Assessment
Culture	Continuous Learning Culture Self-Assessment
Predictability	Solution Train Predictability Measure Program Predictability Measure
Performance	Solution Train Performance Metrics Program Performance Metrics Team PI Performance Report DevOps Health Radar
Lean-Agile Principles and Practices	Enterprise Solution Delivery Self-Assessment Agile Product Delivery Self-Assessment Lean-Agile Leadership Self-Assessment Team and Technical Agility Self-Assessment
Product Progress	Feature Progress Report Cumulative Flow Diagram
Efficiency	Continuous Delivery Pipeline Efficiency Recovery over Time Deployments and Releases per Timebox

Table 2.1.: Overview of the metrics provided by the SAFe framework [25]

2.1.4. Large-Scale Scrum (LeSS)

Next to the Scaled Agile Framework (SAFe), there are several other popular scaling agile frameworks. Large-Scale Scrum (LeSS) is among the most popular scaling agile frameworks, with 4% of respondents in the latest State of Agile Report using it [26]. Even though LeSS is not as popular as SAFe (which is used by 35% of respondents [26]), this section provides an in-depth look into LeSS because multiple of the participants in the case study of this thesis are using LeSS.

Figure 2.3 gives an overview of the whole Large-Scale Scrum framework. LeSS is structured in two general parts [34]. The first part is the smaller "standard" **LeSS** framework, that is intended for use cases with one Product and up to eight teams [34]. The LeSS

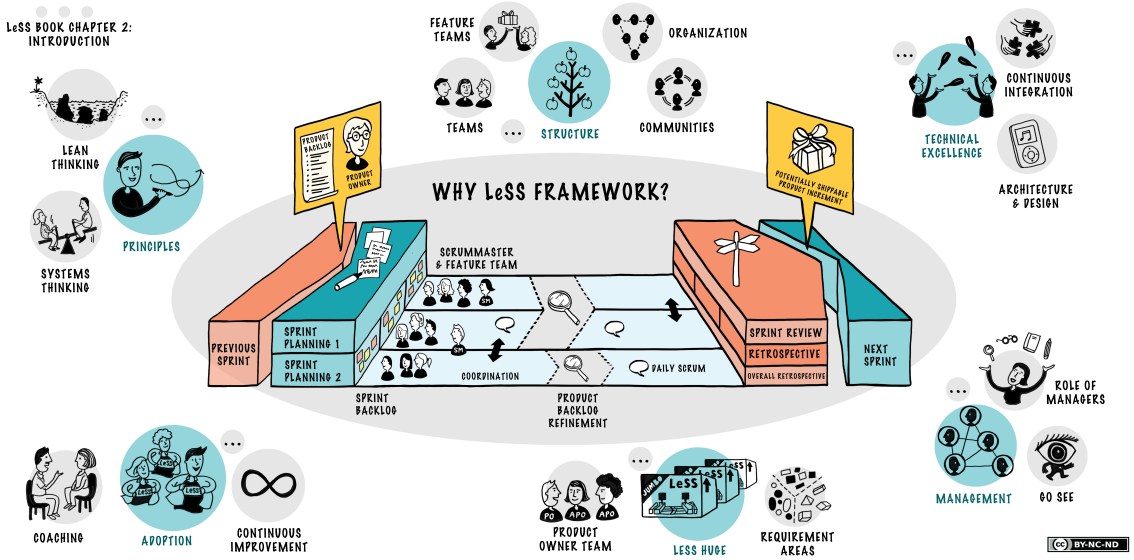


Figure 2.3.: Overview of the LeSS framework, taken from the LeSS website [11]

framework defines the roles of the Product Owner, the Scrum Master, and the Feature Teams [34]. As is the case in Scrum, the Product Owner is concerned with product content matters while the Scrum Master takes care of working methodologies and collaboration improvement [34]. A Feature Team by the definition of LeSS is a cross-functional team that takes care of feature implementation over the whole lifecycle [34]. In LeSS, there is one overall Product Backlog and individual Sprint Backlogs for each of the Feature Teams [34]. Just like in SAFe, Sprints of the teams are aligned. This is called a product-level Sprint in LeSS [34]. Each Sprint in LeSS is prefaced by Sprint Planning One and Sprint Planning Two [34]. Sprint Planning One is done together by all the teams, while Sprint Planning Two is conducted individually by each team. Happening at the end of each overall Sprint, the product Sprint Review is used to share the newly developed product increment with the customers and other stakeholders [34]. After each Sprint an overall Retrospective is done to discuss issues among the participating teams that came up in the individual team Retrospectives [34]. Finally, regular Product Backlog Refinements are conducted with all teams to facilitate understanding and discovery of dependencies across the product [34].

The second part is the bigger **LeSS Huge**, which can be applied to use cases with more than eight teams [34]. A central aspect of LeSS Huge are the so-called *Requirement Areas*. They are defined as the most important groups of requirements by the customers [34]. The Requirement Areas are used to split up the overall product in different areas, which becomes necessary in the large-scale contexts where LeSS Huge is applicable [34]. This split is implemented by assigning each Product Backlog Item to one of the Requirement Areas [34]. Again, like in standard LeSS, all areas operate in a synchronized product Sprint

[34]. Feature Teams are also assigned to individual Requirement Areas and are working according to the standard LeSS framework inside these areas [34].

Goals in LeSS

In contrast to the Scaled Agile Framework, Large-Scale Scrum does not provide explicit guidance on goal-setting. However, it provides information regarding how to structure the different Backlogs used in product development. Based on the principle of Scrum, in Large-Scale Scrum there is only one overall Product Backlog [34]. The Backlog Items that are in the Product Backlog should be "[...] focused around end-to-end customer goals" [34]. In the case of LeSS Huge the overall Product Backlog is complemented with additional Area Backlogs [34]. The Area Backlogs essentially are a decomposition of the Product Backlog by assigning each Backlog Item to exactly one customer Requirement Area [34]. Thus, each Area Backlog can be described as a filtered view on the Product Backlog [34].

Apart from the product-oriented goals, LeSS also promotes the **Go See** management method that has two organizational goals [34]:

- **Better problem-solving capability.** Understand the problems both on the customer and on the team side, and thus increasing the capability to solve them.
- **Improved organizational decision making.** Understand the context that teams are operating in and the impact that (management) decisions outside the teams can have on their work. This will ensure the connectedness of decisions to the actual work environment.

Reporting and Metrics in LeSS

The LeSS framework emphasizes that metrics by themselves cannot be considered good or bad [34]. Instead, the framework recommends that targets and according metrics should be set by the people that are actually using them [34]. Further, each used metric should serve a clearly defined and articulated purpose, avoiding waste by working towards goals with no purpose [34]. However, unlike SAFe, the LeSS framework does not provide a set of metrics that the user can pick from.

2.1.5. Other Scaling Frameworks

Several other frameworks exist for applying agile methodologies at scale apart from SAFe and LeSS. Among the most popular scaling agile frameworks are Nexus, Scrum of Scrums, Disciplined Agile Delivery (DAD), and others [26].

The Nexus framework defines a goal structure similar to the SAFe framework. Each Scrum Team has so-called **Team Sprint Goals** [55]. This is in line with the Sprint Goals in "standard" Scrum. The Team Sprint Goals should be aligned with the Nexus Sprint Goal [55]. The **Nexus Sprint Goal** "[...] is the sum of all the work and Sprint Goals of the Scrum Teams within the Nexus" [55]. It is defined in the Nexus Sprint Planning [55]. Each Nexus

comprises multiple Scrum Teams that develop a common Product [55]. The Nexus works towards the **Product Goal**, which "[...] describes the future state of the product and serves as a long-term goal of the Nexus" [55].

The Nexus framework does not cover the topic of reporting.

We do not elaborate on other frameworks here because they are not relevant in the context of the case study conducted in this thesis.

2.2. Establishing Goals

Section 2.1 described the foundations regarding agile software development methodologies and their application in large-scale settings. Next, this section presents the theoretical background about goal-setting in organizational contexts. This section covers traditional goal-setting theory and how goals are used in software development.

2.2.1. Goal-Setting Theory

Goal-setting theory was originally developed in organizational psychology [39]. It has been developed and empirically tested for over 50 years as of today [38, 39]. In its core, goal-setting theory describes goals as being based on discontent with the current situation and the desire to attain an objective [39].

Mediators between goals and performance	*Goal difficulty*	Harder goals require greater efforts
	Necessary knowledge and skills	Goals require usage of existing skills or to attain new skills
	Task-specific confidence	May mediate other factors such as personality, job autonomy, participation in decisions
	Motivation	The motivation to pursue the goal
Moderators of goal-setting	*Feedback*	Necessary to track progress already made
	Commitment	Viewing the goal as important
	Task complexity	How hard necessary knowledge and skills are to acquire
	Situational constraints	Overload, resource scarcity, etc.

Figure 2.4.: Overview of the mediators of goal performance and moderators of goal-setting by Locke and Latham [39]

The relationship between goals and the individual's or organization's performance in achieving the goals is mediated by several factors, according to the theory. First, goal difficulty influences performance, as harder to achieve goals require more effort and thus lead to increased focus on goal-relevant action [39]. Second, the needed and available skills for attaining a goal are also relevant. A goal may be achieved with existing skills

or knowledge, or require the individual or organization to acquire new ones [39]. Finally, also self-efficacy and motivation to achieve a certain goal are mediating the performance [39].

Apart from those mediators, further influence factors can have effects on performance in goal-attainment. Regular feedback is necessary to be able to evaluate the progress already made [39]. The individual or organization has to actually commit to the goal [39]. Further, task complexity, which depends on how hard it is to acquire the necessary knowledge and skills, influences achievement of a goal [39]. And finally, situational constraints in an organization, such as stress or available resources, can also influence goal-attainment [39].

Goal-setting theory does not discriminate where goals are coming from. In the contrary, it shows that goals are equally effective if assigned by a third party, created participatory, or are self-set [39]. A specific type of goal in goal-setting theory is the **learning goal** [39]. These are goals that focus on acquiring necessary skills or knowledge to be able to achieve further, more complex goals afterwards. These kinds of goals are seen as enhancing *metacognition*, which is the ability to plan, monitor, and evaluate progress towards goal achievement [39]. Metacognition is considered to be of particular importance in environments with minimal structure and guidance [39]. It is worth noting here, that environments with minimal structure and guidance are the intended application scenario for agile development methodologies, as explained in Section 2.1.

Goal-setting for **groups** is also covered by goal-setting theory. It is considered to add additional complexity, because the group's goals might be conflicting with the goals of the participating individuals [39]. Studies have shown that personal goals of individuals that are incompatible with the group goals affect the group performance negatively [39]. Additionally, feedback to individuals in the group tends to shift focus onto individual performance. Only feedback provided on group level has a moderating effect towards group goals (cf. Figure 2.4) [39]. This seems to indicate that group-level performance reporting and feedback is a necessary moderator for group performance.

Finally, on the highest organizational scaling level a clear vision was found to have positive impact on shared goal-setting inside organizations and on achievement of growth goals [39].

Besides goal-setting theory developed by Locke and Latham [38, 39], the management philosophy coined by Peter Drucker is also concerned with management by objectives. In his book on people and performance, Drucker elaborates on management by objectives in organizations [48]. Drucker emphasizes general managers in companies should be focused on business performance, and outnumber functional managers who are concerned with specialized workmanship [48]. Effective management has to primarily "direct the vision and efforts of all managers toward a common goal" [48]. For that end, every employee needs clear objectives that define what they have to contribute to overall organizational performance, according to Drucker [48]. Objectives have to consider both short-term and long-term success [48]. According to Drucker, all managers should define their own objectives themselves, while higher management only accepts or rejects them

[48]. Every employee needs to know the overall company goals to be able to define their own goals [48]. Such an understanding can only be achieved via "communications up" by lower management to upper management [48].

Drucker also emphasizes that every objective should allow managers to measure their performance towards the goal [48]. The measurement result, however, should not be used for control from higher management, but rather for self-control [48]. Using such reports for control from above causes employees to focus on optimizing these reports instead of doing their actual job, according to Drucker [48]. Reports should rather be a tool used by the person that fills them out [48]. Further, Drucker cautions that every measurement has to properly fit into the overall organizational perspective, to avoid managers optimizing towards the wrong measures [48].

2.2.2. Goals in Software Development

Goal-oriented Requirements Engineering

Goals are commonly used in the context of software engineering, especially in the requirements engineering step of projects [51]. In a comparison of several goal-oriented requirements engineering (GORE) methods, Regev and Wegmann [51] give an overview of different definitions and types of goals in software engineering. They find that requirements engineering in general uses goals for "[...] providing the rationale (why) for an envisioned system" [51]. Hence, goals are used to derive concrete requirements [51]. Ideally it should be possible to trace back specific requirements to a goal of the organization. However, the concept of goals is defined differently by existing GORE frameworks [51]. Based on these varying definitions, Regev and Wegmann define goals to refer to an envisioned state of a system, that is formulated by a stakeholder [51]. The compared GORE methods also provide different types of goals. *Achievement Goals* are defined as goals of an organization or system with clearly defined post-conditions for fulfillment [51]. *Maintenance Goals* are goals to *not* enter a specific state, or to keep an existing state constant as is [51]. *Softgoals* are goals without clearly specified criteria of when the desired state is achieved. Thus, they offer room for subjective judgment and interpretation [51]. Additionally, some frameworks also use *beliefs* and *constraints* [51].

Throughout this thesis we aligned our working definition of goals with the definition provided by the Goal-based Requirements Analysis Method (GBRAM):

> *Goals are high level objectives of the business, organization, or system. They express the rationale for proposed systems and guide decisions at various levels within the enterprise.*
>
> — Definition of goals by the GBRAM method [1], as described by Regev and Wegmann [51]

We chose this definition, because we are not focusing on any one specific stakeholder in software development and the definition incorporates the different possible stakeholder viewpoints ("business, organization, or system"). In addition, this definition takes into account different possible scaling levels in an organization at which goals can apply ("various levels within the enterprise"). This is particularly relevant for this thesis, because we are focusing on large-scale organizations that encompass different scaling levels.

In another paper on Goal-oriented Requirements Engineering, van Lamsweerde [64] investigates the use of goals in requirements engineering. They define a goal as an "[...] objective the system under consideration should achieve" [64]. Van Lamsweerde differentiates goals by their *level of abstraction* — high-level, strategic or low-level, technical — and their *type of concern* — functional or non-functional [64]. In contrast to requirements, goals typically require cooperation of multiple agents to achieve them [64]. If a goal can be achieved by an individual agent, it is considered a requirement by van Lamsweerde [64]. Additionally, van Lamsweerde [64] elaborates on logical reasoning about goals, which is not of relevance for this thesis.

Goal Question Metric Approach

The Goal Question Metric (GQM) Approach by Basili et al. [2] is part of the Encyclopedia of Software Engineering [40]. Its focus is on software engineering, but without special consideration of agile methodologies or large-scale settings. The central approach of GQM for progress measurements consists of three parts [2]. First, goals need to be specified for the project to be measured. Second, data from the organization's operation needs to be defined that will be used for evaluating the progress towards the goals. And third, rules need to be specified as to how to interpret the data with respect to the goal [2].

Accordingly, the model used by the GQM approach consists of three layers [2]. A *goal* is defined for a specific object, either a product, process, or resource [2]. Additionally, each goal is defined for a reason and from a point of view by a stakeholder [2]. A goal is always relative to the environment it is defined in, e.g., the current state of the organization [2]. For each goal, *questions* are formulated that describe how the evaluation of goal-achievement is performed [2]. And again for each of these questions data is specified that will be used to answer it [2]. This represents the metric for evaluation of the goal. Basili et al. [2] distinguish between objective data, that is independent from a stakeholders viewpoint, and subjective data, which depends on the object and the viewpoint from which it is collected.

In a subsequent publication, Basili et al. [3] extend their approach to ensure alignment to business. The extension is called **GQM+Strategies** [3]. It separates goals into three categories: *business*, *software-development*, and *project-specific* goals [3]. Strategies serve the purpose of breaking down goals into goals of the next lower level, respectively [3]. Business strategy breaks down business goals by specifying goals at software level, while software level strategy breaks down software-development goals into project-specific goals [3]. At each level a traditional GQM tree is used to operationalize and measure goals [3].

Goals in Large-Scale Agile Software Development

Establishing goals in agile environments is subject to recent research conversation. In a workshop at 2018 International Conference on Agile Software Development (XP 2018), Stray et al. [60] identify lack of clear and common goals as one of the top barriers to autonomous agile teams. Lack of trust within teams also is among the top barriers, causing team members to not commit to team goals [60], while lack of trust between teams and management causes increased demand for reporting and control from management [60].

How goals are considered in scaling agile frameworks was elaborated in Section 2.1.

2.2.3. Types of Goals

Throughout the previous sections we already mentioned several categorizations of goals used by agile frameworks or other publications. This section summarizes another publication focused solely on types of goals.

In a case study with 75 managers of companies from three countries, Bateman et al. [4] created a taxonomy that structures goals along two dimensions: **goal content** and **goal level**. The content of goals is distinguished in ten categories [4]: *Personal*, *Financial*, *Customer*, *Market*, *Operations*, *Product*, *Organization*, *People*, *Competitive*, and *Strategy making*. Further, goals can be relevant to the organization on different levels [4]. *Process goals* are goals that represent the ongoing, tactile operation of business, e.g., continuous learning, interacting with customers, and coordinating employees [4]. *Project goals* are time-bound and focused on a discrete business project, e.g., setting up a new production plant, or hiring a new expert [4]. Next, *strategic goals* focus on differentiation of the organization [4]. They are the basis for resource allocation and often refer to deliverables to customers, quality, or pricing. *Enterprise goals* are goals for the organization as a whole, e.g., growth, or revenue increase [4]. Finally, *ultimate goals* are at the top of the goal-hierarchy [4]. They represent long-term objectives and are of personal or societal nature, e.g., retirement, or improving attractiveness of country as a business location.

The differentiation of goals along the dimensions of content and organizational level is similar to the one made by van Lamsweerde [64] in the requirements engineering domain, as explained earlier. Both authors are considering the level as well as the content type of goals as important factors for categorization of goals. This will be the basis for structuring and categorizing the goals that we observe in the case study in Section 4.

2.3. Reporting Progress

This section explains the theoretical background on reporting progress towards goals. Reporting is important for organizational success. A study by Müller et al. [44] found that establishing reporting on portfolio-level is positively associated with achieving results. Regarding large-scale environments, a shared reporting approach was found to help organizations achieve their goals by channeling information from lower organizational levels

to higher levels [44]. Müller et al. recommend to establish and standardize common reporting practices in an organization, to be able to compare projects on specific metrics [44].

Throughout this thesis we refer to reporting as *quantitative* in cases where metrics form the central aspect, and as *qualitative* in cases where no metrics are used to evaluate status. In the following, relevant literature on reporting in organizations is discussed.

2.3.1. Quantitative Reporting

The ISO/IEC 15939:2007 standard defines a measurement process for software engineering and management in general [24]. The standard focuses on describing central activities necessary to establish, execute, and improve measurements in projects or organizations [24]. Concrete metrics themselves are not provided by the standard.

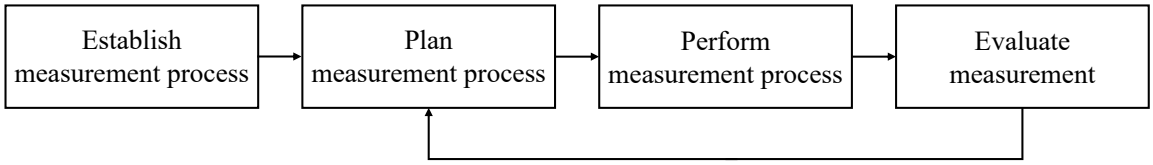


Figure 2.5.: The four core activities of the ISO/IEC 15939:2007 standard as described by the IEEE [24]

The ISO standard splits the measurement process into four separate steps [24]. In a first step, management and employees have to define requirements for the process, and allocate and assign resources needed to carry out the actual process in the later steps [24]. Management and employees have to commit to the defined scope and communicate so. Second, the measurements have to be planed [24]. This entails defining which object is to be measured, which information need is to be satisfied, which measures should be used, and how the data is to be collected, analyzed, and reported [24]. The third step focuses on executing the actual measurements [24]. This means generating and collecting the necessary data on the one hand, and analyzing the data and communicating the results on the other hand [24]. In the fourth and final step of the process, both the results of the measurement analysis and the measurement process itself should be evaluated [24]. Identified opportunities for improvement have to be applied to the product as well as the measurement process [24]. The standard also describes that the four main steps of the process can and should be conducted iteratively to enable adjustment of previously conducted steps to new insights [24].

The Project Management Body of Knowledge Guide (PMBOK-Guide) by the Project Management Institute (PMI) has an extension focused on agile methodologies. The PMI Agile Practice Guide [27] describes — among other aspects of agile methodologies — measurements in agile projects. The Agile Practice Guide recommends usage of metrics primarily to improve the ability to make forecasts and as a basis for decision making [27]. However, the guide emphasizes that empirical metrics in agile — like all predictive

measurements — might not reflect the actual state accurately and experience unforeseen changes [27]. It differentiates between capacity-based and flow-based measurements [27]. Capacity-based measurements use the number of stories or story points to evaluate performance, while flow-based measurements such as lead time and cycle time focus on time and should be used to identify performance bottlenecks [27]. The guide emphasizes that using only story points for measurement violates the agile principle that working software should be the primary measure of progress [27]. They recommend every team to define their own set of measurements, while acknowledging the disadvantage of missing comparability between teams [27]. Finally, the PMI Agile Practice Guide suggests that also Earned Value Management can be applied in agile environments [27]. Earned Value Management (EVM) is a project management technique that stems from industrial engineering and has been transferred to the software engineering domain [21]. The EVM approach consists of ten steps that are summarized in Table 2.2.

Kupiainen et al. [32] conducted a systematic literature review on the usage of metrics in agile and lean software development. In their paper, they review 774 papers and finally select 30 primary studies for detailed analysis [32]. Reasons behind usage of metrics, as well as effects of using metrics in agile and lean software development are their central objectives [32]. The findings of the literature review cluster these reasons and effects of metrics into five key areas [32]. *Sprint and project planning* comprises three activities for which metrics are used: prioritization, scoping, and resourcing. This category contains the fewest identified metrics. The *progress tracking* area contains metrics measuring project progress, visibility, goal-achievement, and workflow balance. *Software quality measurement* metrics measure the level of quality, both directly on the product and indirectly via test-metrics. The *fixing software process problems* category contains the most identified metrics. Metrics in this category are intended to understand problems in the development processes, and to help fix these problems. Finally, the *motivating people* area comprises metrics that are used for motivating employees and to positively influence their behavior. Negative effects on employee behavior, however, are also a possible outcome of applying such metrics, as pointed out by Kupiainen et al. [32]. Employees may cut corners to try to keep the metrics at the desired level [32]. Additionally, Kupiainen et al. [32] present an overview of all identified metrics in the literature, structured according to the taxonomy of software metrics by Fenton and Pfleeger [20]. They differentiate between the *entities* product, process, or resource, and the *attributes* internal or external that are measured [20, 32]. In summary, the most important metrics in agile and lean software development identified by Kupiainen et al. [32] are velocity, effort estimations, customer satisfaction, build status, technical debt, and progress based on working code. Velocity and effort estimations are also the metrics mentioned most often throughout the 30 primary studies of the review [32].

2.3.2. Reporting Practices

This section describes reporting practices that are relevant for the case study of this thesis. The Balanced Scorecard is explained first, followed by Objectives and Key Results.

	Step	Explanation
1	Define work scope	The complete scope of the work has to be defined upfront.
2	Create integrated bottom-up plan	Overall scope has to be broken down into sub-cells, so-called Control Account Plans (CAPs). In sum, the CAPs represent overall progress.
3	Formally schedule CAPs	Definition of what has to be achieved when in the project.
4	Assign Each CAP to an Executive for Performance	Each CAP is assigned the responsibility of an executive.
5	Establish a Baseline that Summarizes CAPs	Definition of how the CAPs are summarized to evaluate overall progress.
6	Measure Performance Against Schedule	The actual performance has to be compared to the planned schedule periodically. A difference between actual and planned state is referred to as schedule variance.
7	Measure Cost Efficiency Against the Costs Incurred	The actually achieved value of the project has to be compared to the actually incurred costs periodically. A difference between the two values is referred to as cost-efficiency factor.
8	Forecast Final Costs Based on Performance	Based on the actual performance, the projects final costs have to be forecasted periodically.
9	Manage Remaining Work	The remaining work has to be managed and planned based on the current state.
10	Manage Baseline Changes	Changes to the scope of the project have to be continuously integrated into the baseline and evaluation approach.

Table 2.2.: Overview of the central steps of Earned Value Management (EVM) for software projects, as described by Fleming and Koppelman [21]

Balanced Scorecard

The Balanced Scorecard (BSC) is a reporting tool developed by Kaplan and Norton [28]. It tries to provide the top-management of an organization with a combination of financial and operational measures of the business [28]. The business is represented from four perspectives in the Balanced Scorecard: the *customer, internal, innovation and learning*, and *financial* perspectives [28]. For each of the perspectives, the BSC holds different goals, and for each goal an appropriate measure is defined [28]. A exemplary BSC is shown in Figure 2.6.

The elements in the *customer perspective* focus on how the organization is viewed by its

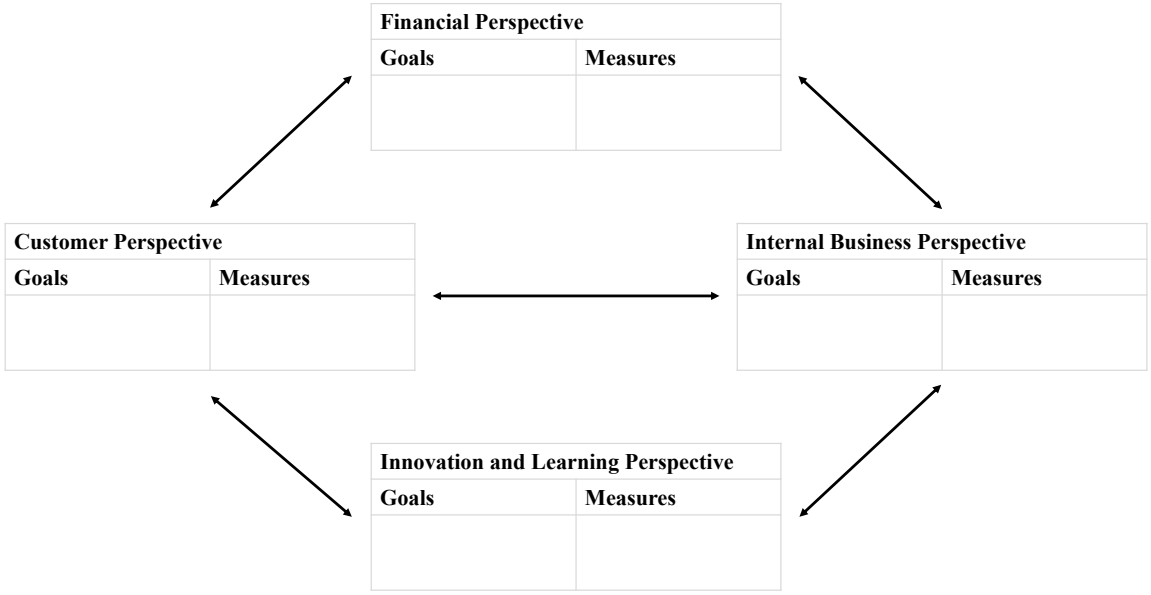


Figure 2.6.: Template for a Balanced Scorecard based on Kaplan and Norton [28]

customers [28]. These typically are concerns and measures of time, quality, performance and service, and cost [28]. The *internal business perspective* focuses on internal operations that are necessary to achieve said customer satisfaction [28]. Thus, here the measures typically revolve around the business processes. Kaplan and Norton emphasize the crucial importance of information systems for aggregating the necessary internal data to do the measurements [28]. In the *innovation and learning perspective*, the BSC focuses on measures to continuously improve products and processes [28]. And finally, the *financial perspective* revolves around measures of the financial success of the organization.

In summary, the BSC is a tool to "[...] translate a company's strategy into specific measureable objectives" [28]. However, the strategy itself is not defined using the BSC [28].

Objectives and Key Results

Objectives and Key Results (OKRs) are a practice to set goals for a specific period of time, e.g., a quarter of a year [67]. As implied by the name the central concepts of the practice are *Objectives* and *Key Results* [67]. An *Objective* is a sentence that should be formulated qualitative and in a way that motivates people [67]. No metric or measurement should be part of an Objective [67]. Further, Objectives should be time-bound with a clear period within which it should be achieved [67]. Objectives should be actionable by the organization / program / team independently [67].

Key Results on the other hand should be quantitative [67]. They are the vehicle to quantify and operationalize an Objective [67]. The number of Key Results per Objective should

be limited, e.g., to about three [67]. Hence each Key Result should define a clear metric for evaluation and a target value for this metric [67].

In general, OKRs should be ambitious for the organization / team, but still be within the achievable range [67]. As a result of this ambitiousness, OKRs might be not or only partially achieved. This has to be part of the mindset of teams, to maintain good spirits amid failure to achieve all OKRs [67]. The OKRs should be established in a focused meeting with the senior management [67]. Before this meeting all employees should be encouraged to raise potential Objectives they think should be pursued by the organization / team in the next quarter [67]. Using these Objectives from employees as well as Objectives of the management, the team in the meeting has to determine the most important ones to pursue in the next quarter [67]. Further, the management team has to define metrics that could be used for measuring the achievement of each selected Objective [67]. These metrics form the basis for formulating the Key Results [67]. Finally, for each Key Result a target value has to be defined that is — as explained earlier — ambitious but still doable [67].

3. Related Work

Chapter 2 presented the theoretical foundations and key concepts that are relevant to this thesis. This chapter discusses several publications with topics that are related to goal-setting and reporting in large-scale agile environments.

Stettina and Schoemaker (2018) [59]

Stettina and Schoemaker [59] conduct a multiple case study in ten large organizations. The focus of the case study is on reporting routines, artifacts, and metrics in Agile Portfolio Management [59]. A particular focus is on knowledge boundaries inside the organizations, and how they are passed by the identified artifacts and routines [59]. Figure 3.1 shows an overview of the findings by Stettina and Schoemaker [59].

Reporting Routines	Reporting Artifacts	Reporting Metrics	Reporting Responsibility
• Cadence-driven • PMO-driven • Tool-driven	• Tool-based • Document-based • Interaction-based	• Performance • Quality • Progress • Status • Contextual	• Development (Team, Architects) • Product (Product / Portfolio managers) • Process (Scrum Master, Release Train Engineer, etc.)

Figure 3.1.: Overview of the findings by Stettina and Schoemaker [59]

In general, Stettina and Schoemaker differentiate between **quantitative** and **qualitative** reporting [59]. Both, quantitative and qualitative reporting are present in the organizations of their study [59]. They find, that qualitative reporting is used to provide context, strategic insight, and possible opportunities, while quantitative reporting is used to quantify progress and value, and to provide tactical insights [59].

In the study, Stettina and Schoemaker identify three types of *reporting routines* [59]. *Cadence-driven* reporting is oriented around development cadence, such as Sprint-length [59]. *PMO-driven* reporting is based on templates and is primarily done manually [59]. These templates may be provided by project management frameworks or the Project Management Office (PMO). *Tool-driven* reporting is facilitated by tools such as Jira [59]. It is done ad hoc and serves day-to-day reporting needs. Further, Stettina and Schoemaker also identify three types of *reporting artifacts* [59]: *Tool-based* artifacts, which are part of tools like Jira, *document-based* artifacts, such as Excel sheets or PowerPoint slides, and *interaction-based* artifacts, which are byproducts of interactions between people [59]. They observe

that usage of document-based artifacts is decreasing with increase of agile maturity of the organization [59]. Finally, they also identify five types of *metrics* [59]. *Performance* metrics, which measure efficiency of work, *quality* metrics, *progress* metrics, *status* metrics, and *contextual* metrics which provide information *on* the work [59].

Apart from these findings on routines, artifacts, and metrics they also observe a division of reporting responsibility into three sections [59]. First, overall product performance reporting is done by product and portfolio managers [59]. Second, reporting on product quality and dependencies is in the responsibility of development teams and architects [59]. Third, reporting on process quality and working methodology is done by Scrum Masters and similar roles [59].

While Stettina and Schoemaker [59] focus on reporting in their study, the perspective on where goals to report on are coming from and how goals are established is not considered. Challenges with the existing approaches are not discussed. Further, they do not investigate the reasons for why the studied organizations apply certain reporting practices or which challenges the practices are addressing.

Lappi et al. (2018) [33]

In a systematic literature review Lappi et al. [33] investigate project governance practices in agile projects. They analyze 42 papers along six dimension of project governance: *goal-setting*, *incentives*, *monitoring performance*, *coordination*, *roles and decision-making*, and *capability building*. For the scope of this thesis, the goal-setting and performance monitoring aspects of project governance are of particular interest. The findings of Lappi et al. [33] in these two categories are summarized in the following.

Goal-setting in agile projects is found to often include close cooperation with customers to establish a shared understanding of project goals and product vision [33]. The most important actors involved in goal-setting are found to be Agile Teams and customers. Lappi et al. [33] describe that a product vision is a frequently used mechanism for goal-setting, because it allows for specific goals to emerge and adapt throughout the product life cycle, instead of fixing goals up-front. Agile projects rely on this product vision and backlogs for goal-setting and coordination [33]. They also find that goal-setting for agile projects typically is described as a continuous, iterative process rather than a separate, up-front step [33]. Further, Lappi et al. [33] note that alignment of project goals with the overall strategy of the organization is discussed seldom in current literature.

For the performance monitoring aspect of agile project governance, Lappi et al. [33] find that iteration reviews are the most frequent reporting practices of Agile Teams, used to demonstrate recent work to customers and collect feedback. They note that the deliverables created in the teams' Sprints are commonly used for overall project progress measurement and monitoring as well [33]. This usage of deliverables for progress measurement and monitoring is attributed to the lack of clear project goals in the earlier phases, since goal-setting is described as continuous and iterative [33]. Consequently, goals are considered to not be suitable to report progress towards, especially in early project phases

[33]. Lappi et al. [33] state that indicators for agile project performance, in contrast to traditional ones, do not focus on individual tasks, activities, and people. Metrics specifically designed for agile environments are found to be discussed only by few papers in the literature review [33]. Further, they find that Scum Masters and Agile Coaches are responsible for team-level performance monitoring, while other measurements are done by the teams themselves [33].

In their summary, Lappi et al. [33] state that agile projects seem to have weaker links to overall organizational strategy than traditional projects, due to the more short-term, iterative approach of agile methodologies. They find that connecting product and project vision to longer-term outcome goals is not discussed in current literature [33]. There is a knowledge-gap on how to connect and align objectives across the different scaling levels of an organization to its strategy [33]. They express the need for further research on this topic. Further, Lappi et al. [33] find that, while multiple papers discussed team autonomy, they did not find literature on how to ensure this autonomy in practice. They conclude that future research on agile project governance should focus on empirical studies to investigate how it is applied on different scaling levels in organizations [33]. Research should be conducted on how agile projects can be connected to organizational strategy via governance practices [33]. This master's thesis seeks to address some of these gaps in current research, as explained earlier in the motivation. Since this thesis focuses on large-scale agile environments, especially the aspects concerning different organizational scaling levels pointed out by Lappi et al. are in scope.

Moe et al. (2019) [42]

In their publication, Moe et al. [42] investigate team autonomy in large-scale agile projects. A particular focus of the study is on goals of teams in large-scale agile, and where these goals are coming from [42]. For the paper, Moe et al. [42] conduct a multiple case study with three companies in the banking and software industries. Overall, they observe 14 teams in three projects using interviews and observations [42]. As part of their key findings, they identify two barriers for team autonomy in large-scale agile [42].

The first barrier is "[...] how goals for the teams are set, what they entail, and how they are communicated to the team" [42]. This is called *overall direction* by the authors [42]. The second barrier "[...] entails how teams coordinate with their environment" [42], which is called *external coordination* by the authors [42].

In the study, they find that shared direction for teams is important in large-scale agile settings [42]. Teams should clearly understand the common goals of the large-scale project, as it is the teams who are in charge of understanding and carrying out the necessary work to achieve the goals [42]. Further, they find that these shared goals "[...] are often set by management without involving the teams, the goals are often equal to deliverables and deadlines, and team members are not always sure what the goals are" [42]. The authors mention that, according to research, motivation and commitment to goals are increased if teams are participating in the goal-setting process [42]. However, the study finds that the

participatory approach is not used in practice [42].

Thus, goals are often not seen as such by the teams [42]. In the study, they identify that teams often interpret goals as delivery deadlines [42]. Goals are seen as merely something to strive for by the teams, because the actual higher-level goals are not communicated to the teams [42]. According to Moe et al. [42], this mismatch of understanding between management and teams is due to the lack of an "arena" for defining shared goals with the direct involvement of the teams. This in turn leads to missing commitment to goals by individuals and teams [42]. It will cause individuals and teams to set their own goals [42]. Hence, the authors emphasize the importance of team-participation in defining goals [42].

The results of the study by Moe et al. [42] are an important input to the research project of this master's thesis. The identified barriers and suggested remedies by Moe et al. inform our answer to the third research question, which seeks to address common challenges in large-scale agile goal-setting and reporting.

Schnabel and Pizka (2006) [53]

Schnabel and Pizka [53] present a process model for software development, called *Goal-Driven Software Development Process* (GDP). The GDP is based on two central design decisions. First, the process focuses on goals over requirements for communication between business and software development [53]. Schnabel and Pizka [53] justify this with the claim that requirements are often elicitated by authors that lack technical expertise, and that development of a software systems may also require adjustments on the supported business functions. Second, the process explicitly incorporates top-down and bottom-up thinking [53]. Goal-definition is driven by stakeholders and business analysts, which constitutes the top-down part of the GDP. Decisions on feasibility and technical details are driven by software developers, representing the bottom-up part of the GDP. Schnabel and Pizka [53] use the term "*top-down thinking and bottom-up acting*" for this combined approach.

The understanding of goals in the GDP is based on the Goal-Question Metric approach by Basili et al. [2] (see Section 2). Goals in the GDP are qualitative, informal descriptions of stakeholder needs, that are supported by a set of questions that describe how goal-achievement should be evaluated [53].

The GDP consists of three general activities that are carried out in iterations [53]:

1. **Goal identification.** Stakeholders and developers collaboratively define goals in small groups. Stakeholders are exclusively responsible for stating the business needs, while developers take care of assessing feasibility and possible technological solutions.
2. **Vertical distribution of tasks.** Goals are assigned to small groups of programmers, which are responsible for deciding how to achieve the goals. The distribution is done by developers themselves.
3. **Implementation and testing.** The software is implemented and tested by developers. Tests should be based on goals, by evaluating the questions assigned to the goal.

While the authors of the GDP clearly reference agile methodologies such as Scrum and eXtreme Programming, they do not explicitly categorize their approach as being agile [53]. They state that the GDP is an iterative and incremental process, with similarities to existing agile and non-agile process models [53]. Further, the approach is not explicitly designed for large-scale use cases [53]. Schnabel and Pizka [53] present the application of the GDP in one case of eight programmers and two business analysts. Reporting is only implicitly considered by the GDP via questions that are linked to goals. Reporting practices or routines are not discussed by Schnabel and Pizka [53].

So, while the GDP is not designed for large-scale agile environments that are of interest to this master's thesis, it serves as a good starting point for structuring the goal-setting process in Chapter 5.

Boerman et al. (2015) [10]

Boerman et al. [10] develop a model for measuring and reporting in agile software development. Their model focuses on measuring quality, progress, and predictions, as well as reporting status on these qualities to external stakeholders that are not part of the actual development [10]. Similar to Schnabel and Pizka [53], Boerman et al. [10] base their model on the Goal Question Metric (GQM) approach by Basili et al. [2]. A clear focus is put on Backlogs as a basis for reporting [10].

The reporting model by Boerman et al. [10] focuses on the goals functional compliance of the developed software system, fulfillment of schedule, optimization of value for money, and minimization of risk of wasted effort. From these goals, they derive several metrics that should be reported on [10]: *enhancement rate*, *scope prognosis*, *project size remaining*, *changed backlog items*, *added backlog items*, *rejected backlog items*, *project size*, *time prognosis*, *estimation shift*, *priority shift*, *backlog items at risk*, *software quality*, *expenses prognosis*, and *effort at risk*. They evaluate the chosen metrics in interviews with an IT program manager, and by applying them to historic data at a case organization [10]. Apart from *changed backlog items*, all metrics were evaluated to be medium to highly feasible and useful by the interviewee [10]. In the case study, the biggest challenge they faced was availability and quality of data to calculate the metrics for reporting [10]. They conclude that data quality for reporting should be considered right from the beginning of a project, and that further research should be conducted to define a set of default metrics for reporting in agile software development [10].

Compared to Boerman et al. [10] the goal of this thesis is not to propose a fixed, concrete set of reporting dimensions and metrics. Rather, this thesis tries to consider and describe reporting routines in general, while also incorporating the goal-setting side.

Cheng et al. (2009) [12]

In their paper, Cheng et al. [12] analyze how monitoring and controlling software development can be achieved using key performance indicators (KPIs) and interventions.

Their goal is to provide a set of KPIs and interventions, that can be used by development managers for any agile methodologies [12]. KPIs are considered to show performance of current methods [12]. According to Cheng et al. [12], management needs to take action — called intervention — whenever KPIs go out of the previously specified desired area. KPIs are used to 'trigger' management interventions [12].

Using a case study with one organization, Cheng et al. [12] identify an initial list of KPIs and interventions used in agile software development. They divide the KPIs and interventions into four different aspects: *team*, *person*, *task*, and *quality* [12]. *Team* KPIs focus on current and future productivity of the development teams, while *person* KPIs focus on individuals' current and future productivity [12]. *Task* KPIs show work progress and spent time, and *quality* KPIs focus on softwae quality with the customer as a primary source of input [12]. Accordingly, the interventions focus on controlling and improving team productivity and collaboration, individual welfare and capability, task management, and awareness of quality [12]. After the initial case study, they validate the KPIs and interventions in three more case studies with software product companies [12]. In total, 24 KPIs and 26 interventions are identified [12].

Cheng et al. [12] state that KPIs should be made visible for the whole organization, to allow other departments to see the current status of the development. Further, they learn that assigning development managers to a fixed team allows them to learn and build up expertise together and, as a consequence, enables more accurate planning [12]. Finally, they recommend to look at the most important KPIs daily in the stand-up meeting. This ensures early discovery of problems and allows to directly discuss potential causes [12]. They conclude that future research is needed on KPIs and interventions, and a method for establishing KPIs and trigger values [12].

As part of this thesis we plan to add to research on KPIs, as demanded by Cheng et al. [12]. However, Cheng et al. do only focus on KPIs and interventions on the reporting side. They do not cover the goal-setting process and their research is not specific to large-scale agile environments.

Dreesen et al. (2020) [18]

Dreesen et al. [18] approach agile software development from a control theoretical perspective. They define *control* in their paper as "any process in which a person or group of persons or organization of persons determines [...] what another person or group or organization will do" [18]. Given this definition, goal-setting can be seen as a control process in cases where goals are not self-assigned. Thus, the paper by Dreesen et al. [18] is relevant to this thesis. We summarize the findings of Dreesen at al. [18] in the following.

Dreesen et al. [18] investigate whether control mechanisms used by agile development methodologies influence team autonomy and team performance. They hypothesize that agile practices likely use different control modes than traditional methodologies, and thus likely have different effects on team performance and autonomy [18].

Control theory differentiates two general types of *formal* and *informal* control modes [18,

P1	Greater use of informal controls positively impacts (a) team autonomy and (b) team performance.
P2	Greater use of formal control negatively impacts (a) team autonomy, while it positively affects (b) team performance.
P3	Greater degrees of an enabling control style positively affect (a) team autonomy, (b) team performance, and (c) control congruence.
P4	Greater degrees of control congruence positively affect team performance.

Table 3.1.: The propositions made by Dreesen et al. [18] on the relation between control modes, team autonomy and team performance

30]. While formal control focuses on defining and structuring what has to be achieved and how, informal control allows for more autonomy of teams (*clan control*) or individuals (*self-control*) [18, 30]. Dreesen et al. also define the concept of *control congruence* as the level of agreement and understanding of a control between controller and controllee [18].

In a first step, Dreesen et al. [18] map 29 common agile practices to the different control modes. Based on their findings in literature and this mapping, Dreesen et al. [18] make four propositions that theorize on the relation between control modes, team autonomy and team performance. The propositions are shown in Table 3.1. The theory is tested using an embedded, multiple-case study [18]. Dreesen et al. find that control practices used by agile methodologies have a positive impact on team performance [18]. They attribute this to a combination of informal controls — which allow teams and individuals to choose methods of goal achievement on their own — and formal controls — which provide a general structure on what to achieve — that are used by agile methodologies [18]. Additionally, they find that a high control congruence positively influences team performance in agile software development [18].

While the work of Dreesen et al. [18] is not focusing on goal-setting or reporting explicitly, their findings on control in large-scale agile are highly relevant for our case study in Chapter 4. Especially their findings on control congruence and on effects of the mixture of informal and formal control mechanisms help us understand challenges and propose potential solutions.

Murphy and Cormican (2015) [45]

Murphy and Cormican [45] conduct a case study to investigate the implementation of a goal-centered performance management program in a large, global software organization. Based on existing literature, they identify five critical success factors for software productivity measurement [45]:

- *Organization.* The measurement program should be based on organizational goals. Each metric has to be tied to a specific goal. Dedicated personnel should be allocated to drive the measurements.

- *Management practices.* Metrics should be introduced incrementally over time. The measurements should be standardized to reduce efforts and be based on an agreement by included departments.
- *People.* All stakeholders have to clearly understand the rationale behind the measurements. Developers should be involved in the design of the measurement program.
- *Information and communication.* The rationale behind all metrics has to be communicated clearly, as well as information on which data is used for what purpose. Feedback mechanisms should be established to show how the collected data is used.
- *Technology.* The collection of data and calculation of metrics should be automated as far as possible. Data should be stored in a repository to enable trend analysis over time.

These success factors from literature are used by Murphy and Cormican [45] to structure the case study they conduct. In the case study, they interview senior managers, middle managers, and software developers [45]. Their findings contain several reasons at the case organization that hinder success of software performance measurement. These reasons include fixation on individual processes instead of an enterprise-wide focus of measurements, project-oriented (i.e., fixed duration) instead of product-oriented (i.e., continuous) understanding of measurements, fear of exposition of failures instead of learning from measurement results, and subjective measurement decision-making and data manipulation due to personal relationships and lack of data audits [45]. Based on these reasons, Murphy and Cormican [45] recommend to implement measurements with a holistic view of the whole organization, to focus measurement at the actual customer value, and to clearly articulate how each management decision relates to overall goals of the organization [45].

The success factors and hindering reasons identified by Murphy and Cormican [45] are relevant for designing the propositions and process models in Chapter 5. However, they focus on software measurement and do not explicitly consider agile or large-scale settings. Their findings thus may only partially be applicable to the case study of this thesis.

Korpivaara et al. (2021) [31]

Korpivaara et al. [31] investigate how scaled agile organizations set performance objectives and metrics. They focus on challenges of performance measurement in scaled agile organizations [31]. For their study they initially derive a set of five performance dimensions of agile development organizations from existing literature [31]. These dimensions are *financial value*, *customer value*, *internal process efficiency*, *collaboration and culture*, and *innovation and learning*. Based on these dimensions they then design their interviews for a case study [31]. In the case study they find that on a higher unit-level in the organization focus primarily is on customer satisfaction and financial value [31]. These unit-level objectives are found to be driven by the overall strategy of the organizations, while they do not find indication for influence of agile methodologies on objectives [31]. On lower levels of the

organizations — program- and team-level — they find that some objectives are cascaded from higher-level objectives, while they also observe a shift in objectives from a focus on *what* towards a focus on *how* [31]. Lower-level performance measurement is found to have a higher focus on efficiency [31]. This difference in focus on performance dimensions is one of their key findings [31].

In general, Korpivaara et al. [31] identify seven performance objectives. These objectives are *productivity*, *time to market*, *quality*, *continuous improvement*, *employee engagement*, *customer satisfaction*, and *alignment* [31]. Further, in their study they document several challenges of measurement in scaled agile programs [31]. These challenges include a perceived disconnect between KPIs and delivered work by teams, influence of external factors on metrics, inadequate data availability and consistency, high efforts in collecting data, lack of skills in setting metrics, and lack of standardization of measurement practices [31].

Korpivaara et al. [31] suggest future research to validate their findings in other organizations and industries. Their findings are highly relevant to this thesis, as we are also investigating challenges and performance measurement in scaled agile organizations. However, compared to Korpivaara et al. this thesis additionally considers qualitative reporting. Goal-setting practices are also only documented by Korpivaara et al. if they have a direct connection to performance measurement.

Karhapää et al. (2021) [29]

The study by Karhapää et al. [29] is not directly related to the topic of this thesis. However, they pursue a similar goal in the area of quality requirements (QRs) management in agile software development. Similar to the goal of this thesis regarding goal-setting and reporting practices, Karhapää et al. [29] seek to understand how QRs are managed in agile development, and why. This thesis, as explained previously, seeks to understand how goal-setting and reporting are done in agile development, and why. Thus, we consider the work by Karhapää et al. relevant as related work, since we can refer to their methodology and research approach for this research project. Karhapää et al. [29] follow the theory building approach by Sjøberg et al. [58], which we also apply in Chapter 5.

In their study, Karhapää et al. [29] conduct 36 interviews in four case organizations to identify challenges as well as practices of QRs management in agile settings. Based on the findings collected from these interviews they derive a theoretical model that conceptualizes how existing practices relate to context factors of the observed organizations, and how existing practices address the identified challenges [29].

Since their concrete findings are focused on QRs management, we do not elaborate about them in detail here. In summary, Karhapää et al. [29] identified 40 challenges in their interviews. However, they note that they did not find any new challenges that were not yet described in existing literature [29]. They structure the challenges in *QR elicitation, QR analysis, specifying QRs, QR implementation,* and *QR validation*. The practices they identify are categorized in *proactive, reactive,* and *interactive* [29].

Berntzen et al. (2019) [8]

In their study, Berntzen et al. [8] investigate how Product Owners (POs) contribute to work coordination in large-scale agile development. Their research is conducted from a coordination theory perspective, based on Gittel [8, 22]. According to coordination theory shared goals, among others, are central to coordination [8]. Berntzen et al. [8] state that overall goals of a large-scale agile development program can be broken down into a goal hierarchy. They state that breaking goals down allows teams to work on their own goals while still contributing to the overall goals [8].

Using a case study approach, Berntzen et al. [8] collect data at a large-scale agile development program using 12 interviews and observation. Their findings indicate that while coordination practices vary between different contexts of POs, high-quality communication is important to establish and maintain shared goals and knowledge [8]. POs contribute to this by frequently communicating both with their team and other POs [8]. They also observe that the coordination practices are changing over time in the case organization, which they attribute to the continuous improvement methods applied in agile settings (e.g., retrospectives) [8]. Finally, they observe that unscheduled conversations contribute to aligning goals, particularly in cases where not all POs have to be involved [8].

The study by Berntzen et al. [8] does not explicitly consider reporting practices. However, their findings on how Product Owners contribute to coordination, goal alignment, and shared knowledge in large-scale agile settings are relevant for this thesis. We build on this knowledge in Chapter 5, where we make propositions on how to address identified challenges.

Dikert et al. (2016) [14]

Dikert et al. conduct a systematic literature review on challenges and success factors in large-scale agile transformations [14]. In their review they document 35 challenges and group them into nine categories [14]. These categories are *change resistance*, *lack of investment*, *agile difficult to implement*, *coordination challenges in multi-team environment*, *different approaches emerge in a multi-team environment*, *hierarchical management and organizational boundaries*, *requirements engineering challenges*, *quality assurance challenges*, and *integrating non-development functions* [14]. Besides the challenges, Dikert et al. also document 29 success factors for large-scale agile development, which they group into eleven categories [14]. While Dikert et al. [14] focus on challenges in large-scale agile transformations, their scope is larger than the one of this thesis. This thesis specifically focuses on goal-setting and reporting in large-scale agile software development, and not on conducting transformations towards large-scale agile setups. Both topics of goal-setting and reporting are not explicitly reflected in the categories of challenges documented by Dikert et al. [14].

4. Case Study

This chapter describes the case study conducted as part of this thesis. First, Section 4.1 presents the objectives of the case study as well as the methodology used to collect data. Section 4.2 gives an overview of the organizational setup of the industry partner of the study. Finally, Sections 4.3 and 4.4 summarize the findings of the case study, with focus on existing approaches at the case organization for goal establishing and reporting towards goals, as well as identified challenges with this current situation.

4.1. Methodology

The case study of this thesis is based on the guidelines and recommended steps by Runeson and Höst for conducting case studies in the context of software engineering [52]. It can be classified as a single-case, embedded study [68]. The *objective* of the case study was to gain an initial understanding of the approaches of goal establishing and reporting towards goals in practice. While the Foundations and Related Work sections of this thesis outline existing literature and frameworks that form the *theoretical frame of reference*, we used the case study to also collect insights from practitioners who are dealing with goals and reporting in their daily work. The case organization from which we sampled the interview participants is described in detail in Section 4.2. Ultimately, the research questions to be answered by the case study reflect the first two overall research questions of this thesis:

- How are goals in large-scale agile software development established and reported at the case organization?
- What are challenges and reasons for establishing and reporting goals in large-scale agile software development?

The selection of the case study methodology is made based on these two research questions and the criteria defined by Yin [68] and Benbasat [7]. These criteria encompass the form of the research questions, the question whether the research requires control of behavioral events, the question whether the research focuses on contemporary events, and the question whether the studied phenomena are covered by an established theoretical base [7, 68]. The research questions both are *how* questions. Our research does not require behavioral control, because it seeks to study a real, productive organization applying large-scale agile software development. We are interested in active organizations, hence the focus clearly is on contemporary events. And finally, large-scale agile development is still a novel area of research lacking an established theoretical base [14]. According to

these answers to the criteria, the case study methodology is selected as the appropriate methodology [7, 68].

Throughout the case study, data source triangulation was achieved by using multiple techniques of data collection [52]. Third degree data was collected in the form of documents, pages from the corporate wiki, backlogs, presentation slides, and others. This data was collected by the researchers and provided by the participating interviewees. First degree data was collected using semi-structured interviews and represents the most important source of data for this study.

Figure 4.1.: The five phases of the semi-structured interviews

The semi-structured interviews followed a structure of four phases. The different phases of the interviews are illustrated in Figure 4.1, and the overall structure of the interviews is shown in Figure 4.2. The interviews started with a structured section containing questions about the personal background of the interviewee, their experience with large-scale agile software development, and their role in the current agile setup. Further, the first section contained questions about the development organization of which the participant is a part of. The second section of the interviews was concerned with the current organization-specific goals pursued by the participant's development organization, as well as the process of how these goals were defined. The questions in this section were open questions, which allowed interviewees to detail the processes currently in place. Finally, the last two sections of the interviews were focused on the reporting procedures used in the development organizations of the interviewees. The third section concentrated on team-level reporting, and the last section on higher-level reporting approaches.

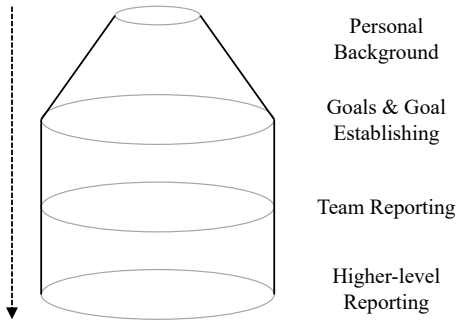


Figure 4.2.: Overview of the structure of the semi-structured interviews, based on the pyramid by Runeson and Höst [52]

Each interview lasted approximately 90 minutes. All interviews, except one (participant asked not to be recorded), were audio-recorded with the consent of the participant

and transcribed afterwards. A preliminary discussion was held with every participant a few days before the actual interview, to ensure a common understanding and definition of relevant concepts of the interview. Further, during the preliminary discussion, a written summary of the research project was handed out to the interviewees to ensure their awareness of how the collected data is used.

All the collected data was analyzed and coded using the qualitative data analysis software tool MAXQDA. The analysis and coding followed the guidelines by Miles et al. [41]. An initial immersion of the researcher was achieved by exploring and reading the whole data set [41]. After this initial immersion, the first cycle coding was conducted [41]. In the first cycle, a descriptive coding technique was applied, assigning codes to significant chunks of data that summarize the chunk in a short phrase or word [41]. Descriptive coding was chosen as it allowed to create an initial inventory of topics that could be used as a basis for the second cycle coding to uncover patterns across all the different data [41]. The codes were created using an *integrated approach* [13, 41]. A provisional starting list of codes was created deductively by the researcher, based on the general structure of the interviews and the relevant concepts of the research project [13, 41]. The individual codes then emerged inductively during the process of analysis and coding, reflecting the encountered concepts and patterns in the data [13, 41]. This mixed approach was chosen because the categories of relevant concepts were already known before data collection, but the actual concepts should emerge from the data itself [13, 41]. After the first cycle, a second cycle coding was conducted [41]. The second cycle built on the inventory of topics created in the first cycle. Recurring, overlapping patterns across the different data were grouped using pattern codes [41]. The artifact built in Chapter 5 is based on these pattern codes that emerged during data analysis.

Interview participants were sampled from four of the major processes of the industry partner organization. The initial sampling was intentional [52] and happened primarily via the Agile Master group inside of the industry partner organization. This group is mainly concerned with work processes and methodology improvement, and is represented across all organizational scaling levels. They are also involved in the goal-setting and reporting processes, hence they served as a natural first sampling source. After the first preliminary discussions with participants were done, we made use of snowball sampling and contacted further potential interviewees who were recommended to us. All the participants of the case study interviews are listed in Table 4.1. Every participant is assigned an alias that is unique and used to refer to the respective interviewee throughout this thesis.

4.2. Case Description

Describing the context of empirical research in software engineering is important to understand the relevance of its findings [19]. Thus, this section describes the context and the participating organization of the case study.

No.	Alias	Role	LSAD experience	Softw. Dev. experience	Program	Duration (h:m)
1	AM1	Agile Master	3 - 5 years	3 - 5 years	A1	1:42
2	PO1	Product Owner	3 - 5 years	11 - 15 years	B1	1:10
3	BE1	Business Expert	6 - 10 years	6 - 10 years	B2	1:29
4	AM2	Agile Master	3 - 5 years	6 - 10 years	B2	1:21
5	PO2	Product Owner	3 - 5 years	11 - 15 years	B2	1:00
6	PO3	Product Owner	3 - 5 years	11 - 15 years	B2	0:52
7	AM3	Agile Master	3 - 5 years	11 - 15 years	B3	1:11
8	STE1	Solution Train Engineer	3 - 5 years	6 - 10 years	C1	no recording
9	LM1	Line Manager	6 - 10 years	6 - 10 years	C2	1:15
10	AM4	Agile Master	3 - 5 years	> 20 years	C2	1:31
11	AM5	Agile Master	11 - 15 years	11 - 15 years	D1	1:24
12	DEV1	Developer, Software Architect	1 - 2 years	6 - 10 years	D2	1:23

Table 4.1.: Overview of the interview participants

The case study was conducted from March to September 2021. The industry partner for the case study is a large German car manufacturer. The organization has well above 100.000 employees, generated a revenue of roughly 100 billion EURO in 2020, and is active internationally. Since the focus of this thesis is in the context of software engineering, the interview participants were located at the internal IT department.

The case organization has been using agile development methodologies for the last 6 to 10 years. Large-scale agile programs have been existing for 3 to 5 years as of 2021. The interview participants gave varying numbers regarding agile and large-scale agile experience of the company, which we attribute to the large size of the overall organization. The time frames named are the most frequently mentioned.

Overall, the organization is structured along major processes that are relevant for automaking. These processes represent important steps in the value chain of the organization. For each process, the value proposition and especially the customer served are different. The IT department of the organization spans across all the aforementioned processes. This is the case because IT services are necessary in all functions and process steps of the case organization.

Inside the different departments of the organization the general organizational structure is standardized and independent from specific agile frameworks. Figure 4.3 gives an overview of the organizational scaling levels at the industry partner. A Domain is the equivalent of a portfolio of Products at the case organization. Hence, a Domain can — but does not have to — contain multiple Products. A Product itself can — but also does

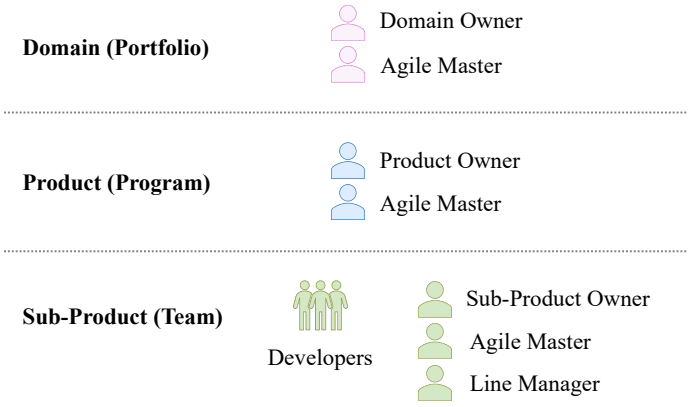


Figure 4.3.: Overview of the scaling levels at the case organization

not have to — consist of multiple Sub-Products. We refer to the Product level at the case organization as Programs in this thesis. And finally, a Sub-Product is the Team level. Each Sub-Product can be developed by one or multiple Agile Teams. At each of the organizational scaling levels an Agile Master is facilitating continuous improvement of the working methodologies. This role is very similar to the Scrum Master role in Scrum but is always present regardless of which agile methodologies or frameworks are applied. Further, a structure of Product Owners is present across the scaling levels. They are in charge of work content and prioritization at the different levels. Finally, the Line Management is responsible for resource and people management. They represent the disciplinary leadership.

Process A

Process A, representing the activities from a product idea to a concrete offering to the end customer, is the first step in the process chain. The activities in **program A1** are concerned with research and development of autonomous driving functionalities for the vehicles produced by the case organization. The overall setup of the program is structured according to the LeSS framework. As interviewee *AM1* told us, the program was set-up together with Craig Larman, the initial author of LeSS. The LeSS program comprises around 100 Agile Teams, 1200 internal employees in total, and is supported by roughly the same number of external employees. Teams are distributed across six different countries. Reporting at program *A1* happens every two weeks on all levels. At team-level, reporting of work progress is done at the end of each Sprint in the Sprint Review meeting. Product-level reporting happens every other week, alternating with team-level reporting. The same principle applies between domain- and product-level reporting.

Process B

The next step, Process B, represents the process from making an offer to receiving an actual order by a customer. The activities in **program B1** are revolving around software and processes for handling the logistics and sales of used vehicles. The program does not use any scaling agile framework, but rather relies on the default vertical scaling given by the organizational structure. It contains 5 Agile Teams with about 40 internal employees, and equally many external employees. Teams are distributed across five different countries. Sub-Product Owners of program *B1* report work progress of their teams at the beginning of each month, while Product Owners report overall Product progress once each quarter of the year. The Domain Owner reports yearly. On all levels, the program is using a standardized template called GMP Sheet. This template is explained in detail in Section 4.3.3.

Program B2 is responsible for the development of vehicle connectivity software and services, mostly with a focus on the off-car components. The program is using SAFe, and consists of roughly 300 internal and 1000 external employees structured in about 80 Agile Teams. Teams are distributed across six different countries. At program *B2*, teams report in the Sprint Review at the end of each Sprint. Products report work progress at the end of each Program Increment (PI), while the overall Domain reports at the end of each Domain Cycle, which is four months at program *B2*.

Finally, **program B3** develops the software needed by dealers to run their business and integrate with the manufacturers' systems and processes. Program *B3* is using the LeSS framework and has about 250 internal and external employees in total, split into 22 Agile Teams. Teams are distributed across four different countries. Every two weeks, at the end of the Sprint, they participate in a Product Sprint Review. All teams report their progress in this meeting. The Product progress is not reported separately, rather the Product Sprint Review with the teams is also the Product's reporting.

Process C

In the third step, Process C contains the activities from receiving an order to delivering the product or service to the customer. The activities of **program C1** are focused on replacing and renewing software systems for all planning- and ordering processes in the company's car manufacturing departments. The program is using SAFe and contains about 120 people structured in 15 Agile Teams. Teams are distributed across two different countries. Every three weeks, teams report their progress in a solution-wide PI Sync meeting. They evaluate each of their PI Objectives on a six-value scale (done, on track, partly, in danger, paused, not started). In a weekly Solution Jour Fixe the Product Owners, Agile Masters, and Solution Train Engineers — who are similar to Domain Agile Masters — assemble an overall Domain reporting. On the one hand, an automated Jira report collects information on how many Backlog Items of the current PI are done, in test, in progress, or open. And on the other hand, Product Owners give a qualitative evaluation statement of their

Product's current status, including a traffic light indication (see Section 4.4).

Program C2 is developing software solutions for the actual shopfloors in the production plants as well as for logistics of car parts. The overall program contains around 1000 people and is using a mixture of practices from both the SAFe and LeSS frameworks. The number of teams was not known by the interviewees. Teams are distributed across five different countries. They report their work progress in a product-wide Sprint Review meeting at the end of each product-cycle. Some milestones are reported in dedicated meetings together with affected stakeholders, in cases of deadlines that are not in sync with the product-cycle. On domain-level progress is reported yearly.

Process D

Finally, Process D bundles financial activities like car financing and leasing. **Program D1** is part of a subsidiary company of the case organization. The subsidiary company is fully owned by the car manufacturer and focuses on mobility leasing and fleet services. Program *D1* is developing a software product that unifies and standardizes the leasing processes for the various mobility offerings of the company across different international markets. The subsidiary company has roughly 2000 employees in total, while program *D1* has 120 employees who are structured in 17 Agile Teams. Teams are located in two different countries. Program *D1* is using the LeSS framework. Every two weeks, at the end of the product-cycle, work progress on program goals is reported in an overall Demo and Sprint Review. Overall progress is evaluated based on automated Jira reports of each team's Sprint results. This is enabled by linking team-level Backlog Items to overall program goals (see linked chain of Backlog Items described in Section 4.3.3).

Program D2 is mostly driven by regulations that the financial department of the automaker has to adhere to. Program *D2* is in charge of implementing and maintaining privileged access management throughout the organization's IT systems to ensure access to systems is only granted to users properly authenticated and authorized. No specific scaling agile framework is used by program *D2*. It employs a staff of about 50 people in total, grouped into roughly 7 teams. Teams are located in four different countries. Reports on team-level are created on a weekly basis and collected by a dedicated person on program-level. Program-level reports are created by this dedicated person based on stakeholders' needs.

4.3. Establishing of Goals

This section details the results of the second part of the case study interviews. Participants were asked to describe their current organization-specific goals, as well as the approach of how these goals were defined in the organization. Together with Section 4.4, this section answers the first and second research questions.

4.3.1. Goal-Setting Process

At the case organization a company-wide **goal management process** (GMP) is used. It is mandatory for all Domains at the case organization, regardless of the used work methodologies. The process was mentioned and explained in seven of the twelve interviews (*PO1, STE1, BE1, AM2, LM1, AM4, PO2*), suggesting that it is highly relevant across the company. It is used to break down goals across all the organizational scaling levels, based on the overall goals for the company defined by the Board of Directors and strategic management circles. The overall company targets are defined using Balanced Scorecards and serve as the input for the GMP. These overall goals are then broken down across the scaling levels by the Product Owner structure that is explained in Section 4.2. This process of breaking goals down was explained by *PO1* and *BE1* to be happening in workshops between the Product Owner at the current level and all the Product Owners from the next lower level. This means a Domain Owner will conduct a workshop with all of the Domain's Product Owners to derive the Domain goals from the GMP goals. The same holds true for the other scaling levels, and Agile Teams are involved in these sessions via their Sub-Product Owner. It was emphasized by *PO1* and *BE1* that by this approach also bottom-up input to the goal-setting is ensured. While the top-down process of defining goals is mostly focused on **what** is set to be achieved, the input from the respective lower level ensures that also the **how** is taken into consideration. These workshops are designed to be conducted in iterations, ensuring top-down goals are aligned with bottom-up input.

While this process was executed once a year on all levels before the introduction of agile methodologies at the case organization, the frequency of execution has changed since then. Today, the overall strategic goals are set yearly, but can span multiple years. On the domain-level Domain goals are derived from the overall strategic goals yearly. On the product-level, the frequency is now differing from Product to Product. Typically, as mentioned by *PO1*, *BE1*, and *LM1*, Products derive goals quarterly from the annual Domain goals. Finally, Sub-Product goals are defined depending on the development cadence of the Agile Teams. For Scrum implementations Sub-Products might derive Sprint goals based on the Sprint length, for example. The cadence of how often goals are set is called a Domain Cycle or Product Cycle, respectively. Figure 4.4 shows how goal-setting using the GMP is implemented in program *C2*.

Despite the very standardized GMP, goal-setting at the case organization is varying between different programs. For program *D1* interviewee *AM5* described an involved process called **dual-track agile goal-setting**. It is used to turn long-term program goals into iteration goals. Hence, it is a concrete example of how program *D1* implements goal-setting in the boundaries specified by the GMP. The goal of the dual-track approach is to ensure participation of the Agile Teams throughout the whole process of goal-setting, especially with a focus on product-related goals. In essence, the dual-track approach is the definition of the next set of quarterly goals while the current quarter of the year is still running. The input to the process are the higher-level GMP goals on the one hand, as well as all additional goals gathered by Agile Teams, customers, partners, or other sources. In the

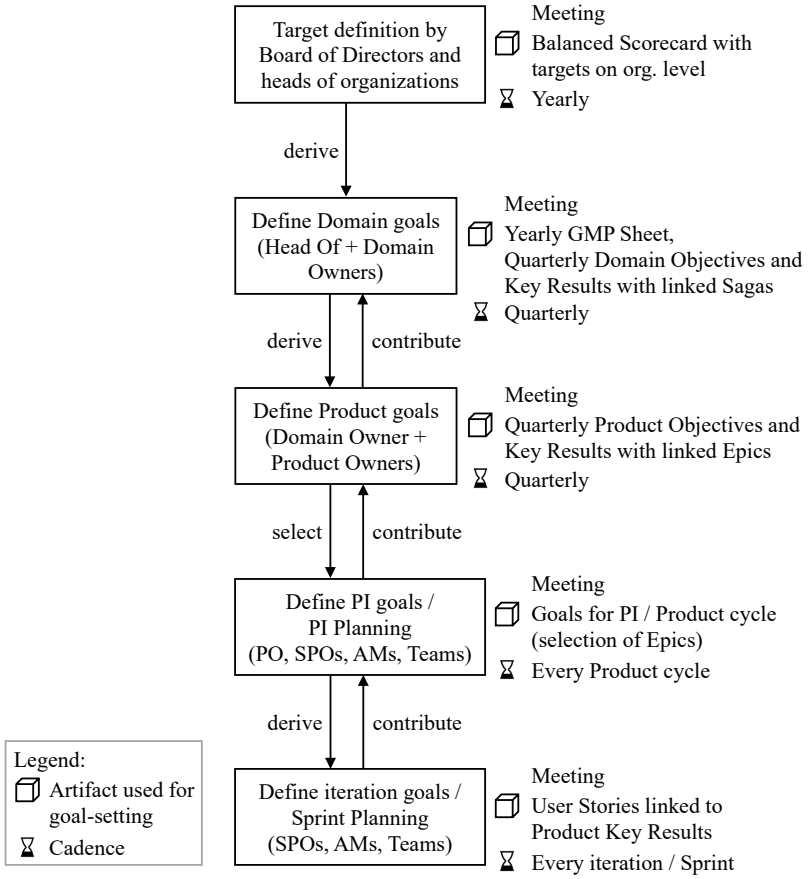


Figure 4.4.: GMP goal-setting as implemented in program *C2*; the notation is based on the one used by Stettina and Schoemaker [59]

first phase, the collected goals are aligned by the Product Owners. If a goal is considered relevant, adequate Sagas are documented in the Program Backlog. Thereafter, Agile Teams conduct a first research and ideation into potential solution directions for the Saga(s). If necessary, a first prototype is created to understand the problem better. The results of this research phase and potential solution directions are documented in Epics in the Program Backlog. In this phase, teams also make a first effort estimation for the Epics together with the Product Owners. Finally before the new quarter is launched, a so-called "*Vision, Roadmap, and Direction (VRD) Focus Day*" is conducted, during which all stakeholders together — lead by the Product Owners — prioritize and plan the Epics for the next quarter based on their goals. Guided by the Epics, Agile Teams are then responsible to define and refine their own Sprint goals throughout the upcoming quarter of the year. This dual-track approach is similar to the process depicted in Figure 4.4. However, it is not using OKRs and contains more involved planning and exploration than the process in Figure 4.4.

4.3.2. Types of Goals

Source	Research Domain	Goal Content	Goal Scaling Level
Bateman et al. [4]	General Management	Personal, Financial, Customer, Market, People, Operations, Product, Organization, Competitive, Strategy Making	Ultimate, Enterprise, Strategic, Project, Process
van Lamsweerde [64]	Software Development, Requirements Engineering	functional, non-functional	high-level strategic, low-level technical
Basili et al. [2, 3]	Software Development	Product, Process, Resource	Business, Software Development, Project-Specific
SAFe [25, 35]	Large-Scale Agile Software Development	-	Solution, Program, Team, Iteration

Table 4.2.: Overview of goal categorizations in related research domains

The goals we collected in the interviews were of different types. To give a holistic overview of all the goals we categorized them using the attributes *content* and *scaling level*. We chose these two dimensions for categorization based on existing literature. Both *goal content* and *goal level* are attributes used by Bateman et al. [4] in their taxonomy of management goals. Further, both dimensions are also present in publications from research areas that we covered in Section 2. An overview of literature using these two dimensions for goal categorization is shown in Table 4.2.

We decided to adopt all three types of goal content from the GQM approach by Basili et at. [2], because they are reflected in our collected data of goals. These are *Product*, *Process*, and *Resource* goals. Additionally, we identified two more types, *Strategic* and *Legal, Security & Compliance* goals, reflected in our data. Table 4.3 provides an overview of the definitions of the identified goal content types. Table 4.4 lists all goals identified in the case study.

In total we identified 51 goals on all organizational scaling levels. The large majority of observed goals is relevant at the portfolio-level (domain; 21 goals) and at the program-level (product; 26 goals). Only a smaller part of the goals is relevant on team-level (sub-product; one goal) and enterprise-wide (seven goals). We also identified that 2 goals are additionally relevant for the line management; these are goals that are concerned with management of employees as a "resource". Further, we find that the majority of the goals can be categorized as *process* goals (22 goals) and as *product* goals (15 goals). Seven goals can be categorized as *strategic* goals, four as *resource* goals, and three as *legal, security & compliance* goals.

We further find that both *process* and *product* goals are roughly equally often pursued on

portfolio- and on program-level. In contrast, Korpivaara et al. [31] find in their study that process efficiency objectives are of higher importance on program- and team-level than on portfolio-level (see summary in Chapter 3). Thus, we note that our findings are not in line with those of Korpivaara et al. [31] regarding the focus of objectives on certain organizational scaling levels.

Goal Content	**Definition of Category**
Product Goals	Goals related to artifacts, documents or deliverables produced during the lifecycle of the offered product or service. Based on Product goals from GQM [2].
Process Goals	Goals related to internal work processes of the organization, typically associated with time. Based on Process goals from GQM [2].
Resource Goals	Goals related to resources of the organization that are used by processes for product development. Based on Process goals from GQM [2].
Strategic Goals	Long-term goals, usually relevant to multiple programs and products, that steer decisions and overall direction of the organization.
Legal, Security & Compliance Goals	Obligatory goals that must be attained; often set / defined by entities outside of the organization

Table 4.3.: Overview of the identified categories of goals at the case organization

Goal	Category	Scaling Level	Source
System stability	Product	Product	*C1*, *B3*, *B2*, *C2*
Legal compliance	Legal, Security & Compliance	Domain, Product	*C1*, *B2*, *C2*, *D2*
Customer value	Product	Product	*B1*, *B3*
Customer satisfaction	Product	Domain	*B2*, *B3*
Service availability	Product	Product	*C1*, *B2*
Transition to cloud	Strategic	Enterprise	*B2*, *C2*
Process efficiency	Process	Domain	*C2*, *D1*
Process standardization	Process	Enterprise	*C2*, *D1*
From specialist roles to Feature Teams	Process	Product	*B3*, *C1*
Release of product	Strategic	Domain, Product	*A1*, *C2*
Transition from car leasing to mobility leasing	Strategic	Enterprise	*D1*
New market entry	Strategic	Enterprise	*D1*
Knowledge transfer and community enablement	Process	Domain	*B2*
Product enablement	Process	Domain	*B2*
Security	Legal, Security & Compliance	Product	*B2*
Scalability	Product	Product	*B2*
Service measurability	Product	Product	*B2*
System performance	Product	Product, Sub-Product	*B2*
Reduce cost	Resource	Product	*B2*
Increase employee engagement	Process	Domain	*B2*
Improve quality	Product	Domain	*B2*
Increase productivity	Process	Product	*B2*
Decrease time to market	Process	Product	*B2*
Improve collaboration	Process	Domain	*B2*
Reduce lead-time	Process	Product	*B3*
Clear definition of rights and responsibilities of each role	Process	Product	*B3*
Serviceability	Product	Product	*B3*
Market share	Strategic	Enterprise	*B3*
Return on investment	Resource	Product	*B3*
From project to product	Process	Product	*B3*
From coordinate to integrate to coordination through integration	Process	Product	*B3*
From independent teams to continuous cross-team cooperation	Process	Product	*B3*
From organizing around technology to organizing around customer	Process	Product	*B3*
From resource thinking to people thinking	Process	Product	*B3*

Competitiveness in the market	Strategic	Enterprise	*B2*
Upgradeability	Product	Domain, Product	*B2*
Always fresh	Product	Product	*B2*
Verfication of product	Product	Product	*A1*
Customer-orientation via user journeys	Product	Domain	*C1*
Reduction of complexity	Product	Domain	*C1*
Replacing old IT system	Product	Domain	*C1*
Reduction of days-to-sell	Process	Domain	*B1*
Product consolidation and harmonization	Strategic	Domain	*B1*
Generate customer leads	Process	Product	*B1*
Reduce days employees call in sick	Resource	Domain, Line Organization	*C2*
Euros spent per employee	Resource	Domain, Line Organization	*C2*
Create a new roles & rights concept	Process	Domain	*D2*
Implement strong authentication	Legal, Security & Compliance	Enterprise	*D2*
Flexibility	Process	Domain	*B2*
Improve structure of requirements	Process	Domain	*B2*
Pursue agile working model	Process	Domain	*B2*

Table 4.4.: List of all identified goals of the initial case study

4.3.3. Goal Definition and Documentation Techniques

As part of the research question on how goals are established, we were also interested in specific definition and documentation techniques used for goal-setting. This section summarizes the findings on techniques used at the case organization for goal definition and documentation.

The standardized Goal Management Process used by the company also comes with predefined templates for goal documentation. These templates were described and shown to us by *PO1*, *BE1*, *AM2*, and *LM1*. The templates are called **GMP Sheets**. They are used for goal definition as well as goal reviewing and reporting. Just like the GMP itself, usage of the templates is mandatory at the highest level — in all observed programs the Domain is documenting and reporting with the templates. The templates are filled by the Product Owner at the respective organizational level. At lower organizational levels usage of the templates varies between different programs. It is not mandatory to use them on lower levels than the domain-level, but some programs (e.g. *B1*) are using them on all levels for the sake of consistency. The templates are differentiating between *change-* and *performance-goals*. Change-goals are referring to necessary changes in the organization to realize the overall strategy. They can be qualitative as well as quantitative. Performance-goals have to be quantitative and represent operational targets that are clearly measurable. In total,

the guidelines of the company recommend to not use more than five goals at one organizational level. For each goal, the name, summary, and target date are documented. For quantitative goals, the target value is documented as well. Typically, the templates are documented using **PowerPoint slides** (described by *BE1, AM5*), but **public Confluence pages** are used as well (described by *PO1, AM5*). The templates are not only used for goal definition and documentation, but also for goal review and reporting. This is explained in Section 4.4.

Another important way how programs document their goals are **Backlogs**. As explained in Section 2, Backlogs are a central tool in several agile frameworks such as SAFe and LeSS. Backlogs are used in every observed program at the company, usually with the help of tools like Jira or CodeBeamer. The most important principle that we observed in the usage of Backlogs (*STE1, AM1, BE1, AM2, LM1, AM4, AM5, PO2, PO3*) is to establish a clearly **linked chain of sub-goals** across all organizational scaling levels. Domains are using highest-level Backlog Items, typically called Saga, that contribute to the long-term Domain goals. Products are using middle-level Backlog Items, typically called Epics, that have to be linked to a Saga they are contributing to. The achievement of the Domain goal directly depends on fulfillment of all linked Saga Backlog Items. The same holds true for Epics and User Stories, which contribute to Product and Team-level goals. This approach is not only used to break down larger goals into smaller chunks, but is equally important for ensuring clear visibility of how work contributes to (higher-level) goals. Interviewee *LM1* stated: *"So, in this approach in this Domain it's pretty straight forward and you can kind of draw a red line directly from the targets from the Board of Directors across all of the hierarchies [...]*". Depending on the Domain, requirements contributing to goals of different levels were documented in the same Backlog or in different Backlogs for each level. Interviewee *AM4* described that program *C2* is using an additional, separate Backlog for organizational development goals and goals resulting from Retrospectives. Further, interviewee *AM2* described that labels are used by program *B2* to cluster Backlog Items into four goal categories: *Customer Functionality*, *Legal, Security & Compliance*, *Stability and Quality*, and *Fit for Future*. The labels are used to give additional structure to the Backlog and allow for filtering.

Most commonly goals at the organization are formulated using the **SMART technique** (Specific, Measurable, Achievable, Relevant, Time-bound). The GMP guidelines demand users to formulate their goals SMART. Interviewees *STE1, BE1,* and *AM5* explicitly mentioned using the technique.

Further, **Objectives and Key Results (OKRs)** are used in four programs (*C1, C2, B2, D2*). Figure 4.4 shows the usage of OKRs in program *C2* in combination with the mandatory GMP elements. The tools used by the organization allow everybody to view OKRs of any Domain, Product, and Sub-Product, if they are used by the respective unit. This is possible because OKRs are documented inside of the organigram. The exact way of implementing OKRs, however, is chosen by the Domain. Interviewee *LM1* gave a detailed description of their program's OKR approach. Based on the yearly Domain targets — which comprise of input from GMP, stakeholders, and Domain-internal topics — quarterly **Objectives** are de-

rived for the Domain. Objectives are formulated qualitatively. This is done by the Domain Owner and the most important external stakeholders. Interviewee *LM1* stated: "*So, it's not like the Domain Owners together with the stakeholders define all Objectives and all Key Results by themselves. [...] That's a big part, to be honest, of the daily work. But there is still room for topics coming up from the teams. [...] They are able to bring those topics up for discussion as some kind of bottom-up input. So, the OKR process is designed in a way that both is possible*". However, no standardized process to incorporate bottom-up input from teams exists. Interviewee *LM1* also acknowledged that bottom-up input to Objectives is considered only up "*[...] to the main department. But not above, probably*". For each of the Objectives, quantitative **Key Results** are defined for the quarter. Based on these quarterly Domain OKRs, each Product derives their own quarterly Product OKRs. On both levels, the number of Objectives is limited to five, and the number of Key Results per Objective is limited to four. Each Key Result has to contain a specific KPI. Backlog Items are then linked to the Key Results. This allows for a quantitative evaluation of how many of the linked Backlog Items are already finished. The Key Result is achieved once all linked Backlog Items are finished. The Objective, in turn, is achieved once all linked Key Results are achieved. Interviewee *LM1* explained the reason for this approach: "*In this Domain we try to assure with this approach that the yearly targets get implemented on a constant basis and the progress is transparent*".

In this section we only discussed the most commonly observed goal-setting practices. To fully answer the first research question of this thesis, all the identified goal-setting practices of the case study are documented in Appendix B.1.

4.4. Reporting towards Goals

The final part of the interviews was focused on reporting routines used to report progress towards the different goals. This section describes how reporting is done at the participating programs of the case organization.

4.4.1. Types of Reporting

In general, across the several scaling levels at the organization, we observed different approaches to reporting. Product, strategic, and legal goals are reported in a "recipient-oriented way" (*LM1, DEV1*) at a recurring cadence. In the case organization this type of reporting is the counterpart to the GMP on the reporting side. It is mostly done via Product and Domain Review meetings, as suggested by scaling agile frameworks LeSS [34] and SAFe [35]. The higher in the organizational structure, the more important this type of reporting is. On domain-level it is mandatory to use the standardized GMP Sheets for this reporting. The GMP Sheets are described in Section 4.3. This reporting is mainly compiled manually (*PO1, BE1, AM2, LM1*). It is done by the Product Owners at the different levels, supported by the development reporting of the Agile Teams on the lower levels.

Process and resource goals, on the other hand, are monitored continuously. The Agile

Masters are responsible for this reporting, and sometimes Line Managers are involved as well. In contrast, this type of reporting is automated to a high degree, e.g., relying on automated Jira Dashboards and tool-generated reports (*AM2, AM3, AM4, AM5*). It mainly serves the purpose of internal information and facilitating planning and continuous improvement.

This observation of a separation of reporting responsibility is in line with the findings of Stettina and Schoemaker [59]. However, in our observations the separation between development and product reporting responsibility is not as clear-cut than it is for process reporting responsibility. This differs from the findings by Stettina and Schoemaker [59]. We rather find that the separation between development and product reporting responsibility becomes more evident the higher the organizational scaling level at which the reporting is done. This is because Sub-Product Owners are involved in team-level development reporting as well as product reporting, while Product and Domain Owners almost exclusively focus on their product reporting responsibility.

4.4.2. Reporting on Team Level

On team-level we observed a primary focus on the continuous, process-oriented and the development-oriented types of reporting. Interviewee *AM5* explained why this is the case:

> *So, in terms of evaluation of the team it is much more important to focus on the process side of things rather than the output side of things. Because output is always a result of how well the team operates, and you can't just force the team to produce more. There is always an underlying reason as to why they are potentially producing less. So, that's what we focus on in terms of reporting on the team level with those quantitative metrics.*

The **Say-Do-Rate** is the most common metric used for reporting at the team-level. It was described to us by interviewees *LM1*, *PO1*, *AM1*, *AM2*, *AM3*, *AM4*, and *AM5*. The Say-Do-Rate is a metric that compares the number of Backlog Items planned for an iteration with the number of Backlog Items actually achieved at the end of the iteration:

$$SayDoRate = \frac{\#ItemsAchieved}{\#ItemsPlanned}$$

As this metric refers to Backlog Items achieved in a given time-frame, it is calculated at the end of this time-frame. On team-level, this is typically the end of each Sprint, but it can also be calculated quarterly or yearly, depending on the time-frame of interest. Further, it can also be calculated per goal, using the Backlog Items that are linked to the goal. The calculation of this metric is done by the Agile Master of the Sub-Product. It is reported to the Sub-Product Owner, who can use this information for planning of the next iteration. This reporting is categorized as quantitative. Interviewees *AM1* and *AM5* explained that the Say-Do-Rate should be preferred over metrics based on Story Points at the team-level.

Story Points should only be used for team-internal discussion and estimation (*DEV1*, *AM1*, *AM4*). The reason for that is to avoid putting external pressure on teams to plan more Backlog Items, and that teams can manipulate metrics based on Story Points quite easily.

Interviewee *AM5* additionally mentioned that the Say-Do-Rate is used to chart a **predictability trend** of a team. The predictability trend shows the development of the Say-Do-Rate of a team over time. It shows whether the team is improving in planning their iterations. Again, this chart is created by the Agile Master to facilitate the Product Owner. Table 4.5 lists all metrics identified at the team-level used by the participating programs.

Name	Description	Program
Say-Do-Rate	Ratio of Backlog Items planned for an iteration and Items actually achieved by the end of it. Calculated at the end of each iteration by Agile Master. Trended out continuously by Agile Master, called Predictability Trend. Formula: $SayDoRate = \#ItemsAchieved/\#ItemsPlanned$	*A1*, *B1*, *B2*, *B3*, *C2*, *D1*
Average Time to Market / Average Lead Time	Average time a Backlog Item takes from creation until finished. Monitored and trended out continuously by Agile Master.	*B2*, *C1*, *D1*
Average Cycle Time	Average time a Backlog Item is in state "In Progress". Monitored and trended out continuously by Agile Master. Formula: $AvgCycleTime = \#AvgWorkInProgress/\#AvgThroughput$	*C2*, *D1*
No. of new Tickets	Automatically generated daily by system.	*B3*, *C2*
Ticket Resolution Ratio	Ratio of closed and newly created tickets in a time period. Automatically generated daily by system.	*B3*, *C2*
Technical Debt Ratio	Shows whether an Iteration created more technical debt than it solved. Calculated at the end of each Sprint by Agile Master. Formula: $DebtRatio = \#DebtItemsCreated/\#DebtItemsSolved$	*D1*
Productivity	Number of finished Backlog Items per time. Calculated and trended out by Agile Team.	*B2*
No. of Deployments	Counted daily by Agile Team.	*B3*
Share of Voice	Share of speaking time occupied by each participant during (Sprint) Review meeting. Monitored by Agile Master Review meeting.	*B3*
Service Availability	Share of time systems were available in a given time frame. Automatically monitored by system.	*B2*
Team Collaboration Happiness	How happy the Agile Team was with working together in the last Sprint. Collected by Agile Master in each Sprint Retrospective.	*D1*
Team Delivery Happiness	How happy the Agile Team was with what they delivered in the last Sprint. Collected by Agile Master in each Sprint Retrospective.	*D1*

Table 4.5.: List of identified metrics at the team-level

Given this focus on continuous, process-oriented reporting on team-level, we still observed product- and development-oriented reporting. The most commonly mentioned means of development-oriented reporting at the case company is the **Sprint Review**. It was described by interviewees *LM1, BE1, PO3, AM1, AM2, AM3,* and *AM5*. In the Sprint Review meetings the Agile Teams, Agile Master, Sub-Product Owner, and sometimes also the Product Owner participate. The user itself was only mentioned by *AM1* as a participant in the Sprint Review. It is conducted at the end of each Sprint, which is typically every one to four weeks at the case organization. In the Sprint Review, the results achieved in the last Sprint are presented by the team to the other stakeholders. This typically includes a live demo of new functionality, as well as a review of the goals of the Sprint. All six interviewees that mentioned the Sprint Review characterized it as a qualitative review.

4.4.3. Reporting on Product and Domain Level

Moving up from the team-level, we observe that the continuous, process-oriented reporting remains of high importance. The reports are highly automated and continuously updated by tools. As on team-level, the Agile Masters are responsible for this reporting. On product- and domain-level the observed programs are focusing on **trend analysis** of several metrics over individual measurements at one point in time. For this purpose, **Value Stream Dashboards** are used in programs *B2* and *B3*, which are based on Data-Warehousing approaches for Jira. This allows for sophisticated analysis of trends and variances in Backlogs over time. The Value Stream Dashboards contain graphs showing the number of Backlog Items currently in progress, the throughput, average lead time and cycle time for each month of the past years, the lead time development for different types of Backlog Items over the past two years, and further trends. Similarly, program *C2* maintains a Product and Team Fitness Tracker on Confluence, where Products and Teams are documenting information on velocity trends, velocity volatility, predictability trends, and the average number of continuous improvement Backlog Items in progress. Table 4.6 lists all observed metrics at product- and domain-level.

Name	Description	Level of Observation	Program
Say-Do-Rate	See description in Table 4.5.	Product, Domain	*A1*, *C2*, *D1*
Lead Time	See description in Table 4.5.	Product, Domain	*B1*, *D1*
Velocity	The sum of completed effort (estimated in Story Points) in one iteration / Product cycle / Domain cycle. Used in automated dashboards.	Product	*B3*, *C2*
No. of Incidents	The number of system outages / incidents in one iteration that affected customer experience. Collected by (Sub-)Product Owner.	Product, Domain	*B2*, *D1*

Estimated vs. Unestimated Backlog Items	Comparison of the number of unestimated and already estimated items in the Backlog. Shows whether the team can keep up with newly incoming requests, especially when tracked over longer periods of time. Used in automated dashboards.	Product, Domain	*B2*, *B3*
Avg. Age of Unresolved Issues by Priority	For each priority category of Backlog Items, the average age of all Backlog Items currently in the Backlog is computed. Used in automated dashboards.	Product, Domain	*B2*, *B3*
Created vs. Resolved Backlog Items	Comparison of the number of newly created Backlog Items and the number of resolved Backlog Items in one cycle. Shows whether the open work is increasing or decreasing over time. Used in automated dashboards.	Product, Domain	*B2*, *B3*
Resolved Backlog Items / Throughput	Number of finished Backlog Items in an cycle. Used in automated dashboards.	Product, Domain	*B2*, *B3*
Work in Progress (WIP)	Number of Backlog Items currently in state "In Progress". Used in automated dashboards.	Product, Domain	*B2*, *B3*
Service Availability	Fraction of time a service is available in an cycle. Collected by (Sub-)Product Owner.	Product	*B2*
Compliance Level	Fraction of applications that have an approved security classification and documented risks in the central risk documentation tool. Collected by (Sub-)Product Owner and Domain Owner.	Domain	*B1*
Security Patch Level	Fraction of applications that are patched within the operating Service Level Agreements (SLAs). Collected by (Sub-)Product Owner and Domain Owner.	Domain	*B1*
No. of implemented Sagas per Domain Cycle	Number of Sagas from the Domain Backlog that have been resolved in a Domain Cycle. Collected by Agile Master.	Domain	*B2*
User Satisfaction	For each new release, stakeholders are surveyed for their satisfaction in the acceptance test / review meeting. Rating on a scale from 1 (not satisfied) to 10 (satisfied).	Product	*D1*
Market Acceptance Test Duration	The duration that the market acceptance test / review meeting took. Longer meetings may indicate need for improvement.	Product	*D1*
Technical Debt Ratio	See description in Table 4.5.	Product, Domain	*D1*
Days to Sell	The number of days it takes for a newly arrived product unit (vehicle) to be sold.	Domain	*B1*
Cycle Time	See description in Table 4.5.	Product, Domain	*D1*

Agile Quality Score	After each cycle, the internal Product Owner and the external development team rate each other on a scale from 1 to 5 stars. Discrepancies in evaluation have to be discussed afterwards. Collected by Agile Master.	Product	*B3*
Capacity	Available capacity (personnel) for an cycle. Especially valuable when interpreting Velocity.	Product	*C2*
Overall Mood	In the overall retrospective, the Agile Master regularly asks each team to rate their current mood as one of the following: "Sunny", "Clear", "Overcast", "Rainy", "Thunderstorm". The responses are tracked over time to identify trends / changes in mood of the teams.	Product	*B3*
Test Line Coverage	The ratio of program code that is covered by tests. Automatically generated by tool.	Product, Domain	*D1*
Test Decision Coverage	The ratio of decision statements in program code that is covered by tests. Automatically generated by tool.	Product, Domain	*D1*
Test Success Rate	The ratio of executed application tests that succeed. Automatically generated by tool.	Product, Domain	*D1*
Avg. No. of Continuous Improvement Process Items Planned	Average number of continuous improvement process Backlog Items that have been planned and done in the last three cycles. Collected by Agile Master.	Product	*C2*
Velocity Volatility	Indicates how much the Velocity changes between the cycles. Low volatility is considered good. Collected by Agile Master.	Product	*C2*

Table 4.6.: List of identified metrics at product- and domain-level (metrics also used on team-level printed in italic)

Interviewees *BE1*, *PO2*, *AM1*, *AM2*, *AM3*, and *AM4* emphasized that a **linked chain of sub-goals** is also important for higher-level reporting. It allows to connect progress made by the teams to the higher-level program goals. This aspect is already explained in detail in Section 4.3 on goal documentation techniques.

Further, on product- and domain-level Agile Masters are continuously employing **automated quality checks** on the Backlog Items. We observed checks of Backlog Items' adherence to the Definition of Ready and Definition of Done. They generate statistics on whether Backlog Items contain proper linkage to parent items, a textual description, estimation in Story Points, definition of acceptance criteria, linkage to addressed defects, and more. Those checks are monitored for Backlog Items in both Domain and Product cycles.

This presence of continuous, process-oriented reporting on all organizational levels fits our observation that *process* goals are the largest part of goals at the case organization (see Section 4.3). It seems intuitive that pursuing many *process* goals poses a demand for process-oriented reporting.

On the other hand, we observed increasing importance of product-oriented reporting

the higher the organizational level. Reporting based on the Goal Management Process (GMP) plays a major role in this category. It was explained to us by interviewees *PO1*, *BE1*, *AM2*, and *LM1*. As with goal-setting, **GMP-based reporting** is mandatory on domain-level, and also frequently used on product-level. For this purpose, the GMP provides standardized GMP Sheets. The GMP Sheets are explained in Section 4.3. Approaches on how these templates are filled differ between programs. The sheets are filled in Domain Reviews, Product Reviews, Area Retrospectives, PI Plannings, and similar events. This largely depends on the scaling agile framework chosen by a particular program. Figure 4.5 shows the variants of GMP-based reporting that we observed.

Scaling Level	Program B1	Program B2	Program C2
Domain	*when*: quarterly and yearly *who*: Domain Owner *how*: GMP Sheet	*when*: half-yearly *who*: Domain Owner, Product Owners *how*: workshop to fill GMP Sheet	*when*: yearly *who*: Domain Owner, Stakeholders *how*: discussion to fill GMP Sheet, using OKRs that were derived from GMP goals
Product	*when*: quarterly *who*: Product Owners *how*: Confluence page, autonomously filled by POs with traffic lights per goal	*when*: quarterly / end of PI *who*: Product Owners, Sub-Product Owners *how*: Confluence page	*when*: deadline of Objective / milestone *who*: Product Owners, Sub-Product Owners *how*: Key Results linked to Objective
Sub-Product	*when*: monthly *who*: Sub-Product Owners *how*: Confluence page, autonomously filled by SPOs with traffic lights per goal	*when*: end of Sprint *who*: Sub-Product Owners, Teams *how*: Sprint Review	*when*: deadline of Objective / milestone *who*: Sub-Product Owners, Teams *how*: Key Results linked to Objective

Figure 4.5.: Overview of observed variants of GMP-based reporting

A frequently used means of reporting at the company is the so-called **Traffic-Light Reporting** (*PO1, AM1, AM2, LM1, BE1, PO2, PO3*). It entails assignment of a color — green, yellow, or red — to a goal, which signals the current achievement status. A green color signals that progress towards goal-attainment is on track as planned. Yellow signals problems that are deemed resolvable by the reporter. The red color code is assigned in cases where the reporter does not see possibility to autonomously resolve problems and is seeking help. This type of reporting is a mandatory part when filling the GMP Sheets, but is also used independently of the GMP. We observed two ways of how the appropriate color codes are derived. On the one hand, reporters may assign a color code based on their qualitative judgment and discussion with stakeholders. This is suitable for non-quantifiable goals. Programs *B1* and *B2* are exclusively applying this approach. On the other hand, the color code may be based on quantitative metrics and defined thresholds for each color-code. Program *A1* is using the Say-Do-Rate as a basis for traffic-light reporting of team progress to the Product Owners. Program *C2* is assigning color codes to Key Results in their OKR setup. They are using the Say-Do-Rate as well to select colors for Key Results based on their linked Backlog Items. Based on how the color codes are derived, the traffic-light

reporting can be categorized as qualitative or quantitative.

Summary on usage of metrics for reporting

In total, we observed the usage of 12 metrics on the team-level, 22 metrics on the product-level (program-level), and 16 metrics on the domain-level (portfolio-level). Metrics therefore seem to be of common use for reporting and measurements on all scaling levels at the case organization, with a tendency towards usage on product-level (program-level).

In this section we documented all the observed metrics, but we only discussed the most commonly observed reporting practices. To fully answer the first research question of this thesis, all the identified reporting practices of the case study are documented in Appendix B.1.

4.5. Challenges and Reasons

To answer the second research question, we also investigated the reasons and challenges behind the current approaches of goal-setting and reporting at the case organization. In total we collected 19 challenges, 14 reasons behind the selected reporting approaches, and 13 reasons behind the selected goals. While not all of them are specific to large-scale agile environments we decided to document them anyway to ensure a holistic portrayal of the current situation. A list of all identified challenges is shows in Table 4.7, a list all identified reasons for the current reporting approaches in Table 4.8, and a list of all identified reasons for the current goals in Table 4.9. The challenges are used to define the objectives for the model approach developed in Chapter 5. Challenges that are specific to large-scale agile environments are printed italic in the table. In this section, we elaborate on the most common challenges and reasons.

Challenges

A very frequent challenge we encountered are **prioritization conflicts between different goals** (C2). This is clearly a goal-setting challenge, however it is not strictly specific to large-scale agile software development. It was explained by interviewees *LM1*, *AM1*, *AM4*, and *AM5*. Prioritization conflicts may arise from different goals set by stakeholders, goals set by the program itself, or rules that limit the number of allowed goals to be set — as implemented by the OKR approach in program *C1*.

ID	Name	Description	Category	Interviewees
C1	*External dependencies limiting autonomous goal-setting*	External dependencies like stakeholder commitment to goals, dependencies on other products or teams make it hard to define clear goals that can be achieved autonomously by the Product / Domain.	Goal-Setting	*PO1*, *STE1*, *LM1*, *PO3*
C2	*Prioritization conflicts between goals*	Balancing different needs of stakeholders.	Goal-Setting	*LM1*, *AM1*, *AM4*, *AM5*
C3	No goals for individual employees	Reporting on individual employees is not allowed by workers union or too resource-intensive in large programs.	Goal-Setting	*PO1*, *STE1*
C4	*Management control limits team autonomy*	Fixed yearly / quarterly goals limit autonomy of Product Owners and Agile Teams. Reporting demanded by management is perceived as intrusive by Agile Teams.	Goal-Setting, Reporting	*AM4*, *AM5*, *STE1*, *DEV1*, *PO2*
C5	*Unclear goals from higher levels*	Goals received from higher org. levels are not clearly defined and explained to all stakeholders.	Goal-Setting	*AM2*
C6	Define current state and target state for qualitative goals	For qualitative, non-measurable goals it is often hard to clearly define the current state and the target state to be achieved.	Goal-Setting	*AM2*
C7	*Missing attachment of teams to goals*	Agile Teams lack attachment and commitment to goals they did not define themselves.	Goal-Setting	*AM4*
C8	Too rigid fixation on goals	Focusing on goals causes lack of appreciation for necessary routine / operational work.	Goal-Setting	*AM3*, *LM1*
C9	*External contracts limiting Feature Team working model*	External contractors are not allowed to operate as cross-functional Feature Teams. Instead, they own specific components only.	Goal-Setting	*AM3*
C10	Missing link between organizational goals and realization	Organizational goals often do not clearly define what is to be done and implemented to achieve the goal.	Goal-Setting	*AM3*
C11	*Resource constraints for goal-setting and reporting*	The goal-setting and reporting processes consume a lot of resources.	Goal-Setting	*BE1*, *PO3*, *DEV1*
C12	Reporting qualitative goals is arbitrary	Reporting qualitative goals is based on "arbitrary" judgment of individuals and allows for green-shifting.	Reporting	*PO1*, *AM1*, *AM2*, *AM3*
C13	*What to report on higher levels*	Higher-level reports often lack focus. They either report too detailed or lack necessary information to identify problems. It is hard to define what is (not) relevant to be reported on higher levels.	Reporting	*AM1*, *LM1*, *PO3*
C14	*Missing link to higher-level goals*	Progress reports do not clearly show how it contributes to goals on higher levels.	Reporting	*PO1*, *STE1*
C15	*Missing automation*	Assembly of reports is often done manually, which consumes resources.	Reporting	*PO1*
C16	*Gathering data*	Collecting data for reports is resource-intensive.	Reporting	*PO1*, *BE1*

C17	Conflict of interest hindering objective reporting	Conflicts of interest caused, e.g., by office politics or bonus payments connected to reporting outcomes, hinder objective reporting.	Reporting	*PO1*, *BE1*
C18	Cost-benefit trade-off of reporting	It is hard to determine how much reporting is needed to provide necessary information while minimizing effort and cost.	Reporting	*BE1*, *PO2*, *PO3*
C19	*Reporting delayed due to hierarchy*	Due to hierarchies in large-scale organizations, reports often reach higher levels with a certain time delay.	Reporting	*PO2*

Table 4.7.: List of all identified challenges of the case study (challenges specific to large-scale and / or agile settings printed italic)

Another common challenge is the **limitation of team autonomy by management control** (C4). Fixed goals defined by higher-level management can limit autonomy at lower levels. This was phrased by interviewee *AM4* as follows:

> *I am struggling with the advantages of agility on the team level sometimes, because we tell the teams what they have to do within one quarter, and then we break it down into four Sprints of three weeks. And actually the four Sprints of three weeks they are predefined completely because you know what you have to do. So, agility from one Sprint to another is not really necessary.*

This effect not only applies to the Agile Teams but it also restricts the (Sub-)Product Owners in their ability to autonomously prioritize the Backlog in their area. Further, stakeholders that expect accurate deadlines and effort predictions can be another cause for this problem. It forces programs to stick to deadlines instead of allowing for necessary adaptation of schedules and team-priorities. On the other hand, several interviewees also mentioned that reporting demands are perceived by the teams as management control, while not being the actual intention behind such demands. This challenge was discussed by interviewees *AM4*, *AM5*, *STE1*, *PO2*, and *DEV1*. It is both a goal-setting and reporting challenge.

Next, **reporting progress towards qualitative goals** (C12) is also a common challenge. Qualitative in this context refers to goals that cannot be clearly measured and that do not come with defined thresholds for achievement. In such cases, the person tasked with compiling the report often has to base the evaluation of goal-progress on discussions or their individual judgment. This is identified by interviewees as a frequent source of seemingly arbitrary reports. Interviewees also mentioned green-shifting along the reporting hierarchy. Problems reported at lower levels were displayed less concerning on higher levels, because reporters on higher levels deemed the problems less serious. This is a reporting challenge that was named by interviewees *PO1*, *AM1*, *AM2*, and *AM3*.

Reasons

In Section 4.4 we described the observation that product-oriented reporting is of increasing importance on higher levels. The reason behind this observation, which is also overall one of the most commonly named reasons, is in stakeholder-orientation. As explained by interviewees *AM4* and *AM5*, external stakeholders are primarily interested in development output and product progress (R2). Because higher-level reporting — in contrast to team-level reporting — is also done for external stakeholders, this may explain the shift of importance of product-oriented reporting. On the other hand, process-oriented reporting is less relevant for external stakeholders according to the interviewees, because external stakeholders often do not care about internal development processes and work methods. Another reason, based on which interviewees *PO1* and *PO2* selected reporting practices, are organizational regulations and governance (R12). Depending on the process or portfolio that a program is part of, often there are overarching regulations by the governance department that impose specific reporting practices to be used by all programs. The reasons behind the identified reporting practices are documented in Table 4.8. Based on our observations we structured the reporting reasons into four categories. *Information needs* describe that certain reporting practices are applied by interviewees because they had a concrete need for information to be addressed. *Agile values and principles* describe that reporting practices are selected based on agile values. *Regulations* describe that some reporting practices are chosen because of regulations that influence how reporting can or has to be done. *Size of program / organization* describes that interviewees applied certain reporting practices because of the size of their agile development environment.

A commonly named reason behind the identified goals are customer requests (*AM1, AM2, PO2*). Interviewees explained that they are pursuing goals because customers requested them to move into a certain direction. Intuitively, this makes sense because ultimately the goal of any large-scale agile development program is to satisfy the needs of the customers. Another reason, named by *PO1* and *AM5*, is the organizational strategy. Interviewees explained that they derived goals for their programs from the overall strategy of the organization. By doing so, they ensure that the organizational strategy is reflected in the program goals and is actually implemented by the program's daily work. All identified reasons behind the identified goals are listed in Table 4.9. Again, based on our observations we grouped the reasons behind the goals into four categories. *Internal needs* describe that goals have been derived from internal needs and desire. *Customer needs* describe that goals have been set based on requests from customers towards the large-scale agile development organization. *Strategy of organization* describes that certain goals are pursued based on the overall strategy of the organization. *Regulations* describe that goals are driven by regulations that the organization has to comply with — thus such goals are not optional but obligatory to pursue.

ID	Name	Description	Category	Source
	Information Needs			
R1	Type of reporting depends on goal and stakeholder	The suitable reporting is chosen based on the type of goal and stakeholders' information demands.	Team Reporting, Higher-level Reporting	*AM4*, *PO3*
R2	External stakeholders are mostly interested in product and program progress	Product-oriented reporting becomes more important on higher levels, because external stakeholders are more interested in product progress than internal processes.	Higher-level Reporting	*AM5*, *AM4*
R3	Learning by doing	No precedent existed at the organization. The current approach emerged by experimentation.	Team Reporting, Higher-level Reporting	*STE1*
R4	Reporting should provide needed information to plan next iteration	The information needs to plan the next iteration influence what should be reported in the current iteration.	Higher-level Reporting	*AM5*
R5	Understand team performance	The reporting was implemented to gain understanding of how teams perform.	Team Reporting	*DEV1*
R6	Higher management only interested in quantitative summary	Higher management is only interested in an aggregated, quantitative summary.	Team Reporting	*PO3*
	Agile Values and Principles			
R7	Avoid putting pressure on Agile Teams	Say-Do-Rate reporting is chosen to minimize external pressure on Agile Teams.	Team Reporting	*AM1*
R8	Organizational goals are never really done	Measurements of organizational goals are trended out because they only indicate improvement relative to time.	Team Reporting	*AM3*
R9	Reporting based on scaling agile framework	The chosen reporting method is suggested by the scaling agile framework in use.	Team Reporting	*AM5*
R10	Working process is most important on team-level	Focus is on working process at team-level, because output is result of how well teams are working.	Team Reporting	*AM5*
	Regulations			
R11	Restrictions by workers union	Regulations by the workers union prohibit reports lower than team-level.	Team Reporting	*AM4*
R12	Organizational regulations and governance	Reporting is regulated and standardized (partially) by the organization, especially the GMP.	Higher-level Reporting	*PO1*, *PO2*
	Size of Program / Organization			
R13	Product-goal reporting on team-level gives only isolated information	Focus is on working process at team-level, because team-level reports lack program context and only provide isolated information.	Team Reporting	*AM5*
R14	Large-scale programs rely on automation	With increasing program size automation of reporting becomes more important.	Higher-level Reporting	*STE1*

Table 4.8.: List of all identified reasons for reporting approaches in the case study

Name	Description	Source
Internal Needs		
Metric indicates need for improvement	An internally monitored metric indicates need for improvement because of currently bad measurement values.	*AM2*
Operational misalignment between markets	Processes and tools between markets are differing, causing portability and efficiency issues.	*AM5*
Legacy IT systems	Legacy systems and applications, and modernization needs influence goal-setting.	*STE1*
Changing internal processes	Changes to internal processes and structures influence goal-setting.	*STE1*
Integration with other parts of company	Products and organizational structures have to be integrated and aligned.	*AM1*
Goal is quantifiable	Preference is given to goals that allow for quantitative evaluation.	*PO1*
Missing flexibility and speed	The organization was not flexible enough to deal with the changing environment, and thus pursued agile methods.	*PO3*
Customer Needs		
Customer requests	Goals are derived directly from customer requests.	*AM1*, *AM2*, *PO2*
Increased orientation on customer value	The agile transformation increases focus of goal-setting on customer value	*AM3*
Changing customer needs	Changing ways of working and needs of customers influence goal-setting process.	*AM5*
Strategy of Organization		
Goal as part of strategy	Goals are directly derived from the overall organizational strategy	*PO1*, *AM5*
Changing company priorities	Changes to the overall company priorities (e.g., electrification, sustainability) influence goal-setting.	*STE1*
Regulations		
Governmental regulations	Governmental agencies (e.g., TÜV) and regulations influence goal-setting.	*AM1*, *DEV1*, *PO2*

Table 4.9.: List of all identified reasons for the specific goals pursued in the case study

5. Theoretical Model

This chapter formalizes our findings from literature and our initial case study. Based on these findings, we theorize on important propositions for goal-setting and reporting in large-scale agile software development. Using this theory, we then develop a model approach for goal-setting and reporting in large-scale agile environments. The propositions and the approach represent the core artifacts of this thesis, and are developed to answer the third research question. They also aim at providing guidelines to practitioners on how to establish goals and implement reporting routines in large-scale agile software development. Further, they explain possible actions to avoid and mitigate common challenges that may arise in this context. Our model approach is not intended to replace established scaling agile frameworks, it rather seeks to offer additional guidance independent of the exact framework that might be applied. This chapter is structured as follows: Section 5.1 describes the methodology used to build the theory. Then, Section 5.2 explains the relevant constructs and propositions for our approach. The concrete process model derived from the constructs and propositions is discussed in Section 5.3.

Throughout this chapter the text and figures will reference challenges and reasons identified in the previously presented case study. Challenges, which are addressed by a certain design decision of our model, are referenced using the IDs from Table 4.7 starting with "C" (e.g., C1, C14). Reasons, which a design decision of our model is based on, are referenced using the IDs from Tables 4.8 starting with "R" (e.g., R1, R5).

5.1. Methodology

To formalize and structure the findings from the previous sections we follow the guidelines by Sjøberg et al. [58]. The guidelines describe recommended steps for theory-building in software engineering. Because this thesis focuses on large-scale agile software development, these guidelines are applicable. Sjøberg et al. differentiate five types of theories [58]. The goal-setting and reporting approach developed in this chapter can be classified as *design and action* theory, because it describes "how to do things" [58]. Karhapää et al. [29] develop a similar theoretical model using the framework by Sjøberg et al. [58] focusing on management of quality requirements in agile software development. While not dealing with the same topic, we reference Karhapää et al. [29] for methodical guidance on using the framework by Sjøberg et al. [58].

Following Sjøberg et al. [58], a theory in software engineering comprises entities of the types actor, technology, activity, and software system. In the theory, an actor can apply

technologies to carry out activities on the software system of concern [58]. Actors can be individuals, teams, or even the whole organization [58]. Technologies are methods, techniques, or tools that are used to carry out an activity, which in turn can be the planning, building, or analysis of a software system [58]. Additionally, a theory also contains propositions [58]. Propositions represent interactions between constructs, and influence that one entity in the theory has on another entity.

5.2. Model Constructs and Propositions

This section presents all the entities and propositions that are part of the model. These entities and propositions are based on our findings from literature and the case study, and form the basis for the suggested approach that is described later in this thesis. This section describes the actors, which activities they are executing, and which technologies they are using for their activities. Tables 5.1 and 5.2 list and summarize all the propositions that are discussed in the following. General propositions formalize general design decisions for the goal-setting and reporting approach. Mitigation propositions on the other hand are intended to address the identified challenges from the case study. They represent practices that should be applied to address specific challenges. Thus, the mitigation propositions are central to answer the third research question of this thesis.

Since this thesis is concerned with goal-setting and reporting in large-scale agile software development, this can be described as the scope in which our theory and approach are applicable. Hence, only constructs and propositions relevant to goal-setting and reporting in large-scale agile software development are part of the model. As proposed by Sjøberg et al. [58] we also present our model using their UML-like notation.

5.2.1. Actors

Our model approach to goal-setting and reporting comprises the actors Agile Team, Agile Master, Product Owner, and Line Manager. All of these actors are part of the overall organization, which itself is modeled as an actor. The organization has different scaling levels at which actors can be operating on. Dividing the organization into scaling levels or areas is common to scaling agile frameworks (cf. [25, 34]). For consistency reasons, we adopt the terminology used in the case study, which contains Team, Program, Portfolio, and the overall Organization as scaling levels. Figure 5.1 gives an overview of the actors in the model. This selection of actors is influenced by roles and responsibilities in several popular agile frameworks such as Scrum, SAFe, LeSS, and Nexus [25, 34, 55, 56], as well as on the observations at the case organization. While Agile Masters and Product Owners are usually present on all scaling levels throughout the organization, Agile Teams only operate on team-level. Line Managers are operating on all levels if necessary, while typically being assigned to one team or program. Each actor has a reporting and goal-setting responsibility, which are the central and only responsibilities that are in scope of our model.

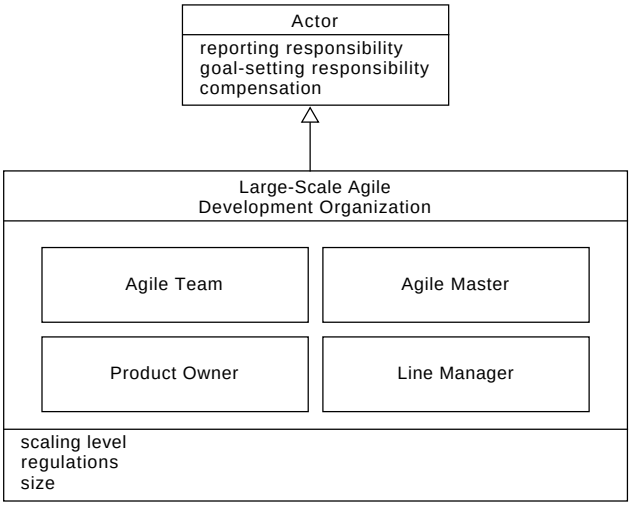


Figure 5.1.: Overview of the actors in the theoretical model

The reporting responsibility is modeled based on our findings in Section 4.4, and is in line with Stettina and Schoemaker [59]. Reporting responsibility can be of type development, product, or process [59]. Development reporting responsibility in our approach is placed with the Agile Teams and Sub-Product Owners. As Agile Teams operate only on team-level, we found that development reporting is of particular relevance at lower levels, while product and process reporting are relevant throughout all scaling levels (see Section 4.4). Product reporting responsibility is placed with the Product Owner hierarchy (i.e., the Product Owners at the different scaling levels) in our approach, and process reporting responsibility with the Agile Masters and Line Managers. On team-level our model suggests to mainly focus on development and process-oriented reporting, based on our case study findings that external delivery pressure on teams should be minimized (R7, R10, R13). Product-oriented reporting should be the focus on program- and portfolio-level, because external stakeholders that receive the overall reporting are usually mostly interested in product and program progress (R2), not in internal working processes of the program. Hence, we propose **P1**: *The scaling level at which an actor operates influences their reporting responsibility and the necessary type of reporting*. Further, the correct type of reporting not only depends on the scaling level, but also on the stakeholders that an actor is reporting to (R1, R2, R4). We propose **P2**: *The recipient of the report influences what should be reported and by whom*.

Goal-setting responsibility, on the other hand, is structured according to the goal content categories we identified in Section 4.3. In contrast to reporting responsibility, goal-setting responsibility for certain types of goals is not clearly mapped to specific actors in our model. We find in the case study that goals are based on internal needs, customer needs, overall strategy of the organization, as well as regulations (see Table 4.9). Interviewees (*LM1, AM5*) explicitly mentioned that goals of several categories could be coming from different actors (see Section 4.3). Thus, in our approach we propose **M1**: *Goal-setting responsibility should be shared among actors, to facilitate collaborative goal-setting practices*. Instead of exclusive goal-setting responsibility of specific actors for certain types of goals, all actors should be able to contribute and propose goals of any type. The shared responsibility of all actors gives the required freedom for collaborative goal-setting activities and practices. Collaborative goal-setting addresses the challenges of missing attachment of Agile Teams to goals (C7) and receiving unclear goals from higher levels (C5), which we

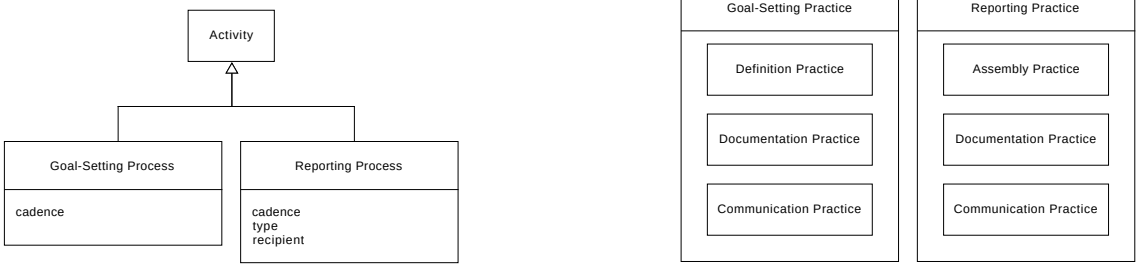
identified in the case study. This design decision is influenced by findings from literature, that shared goals and collaborative goal-setting are important for effective coordination in large-scale agile software development [8, 42]. It is also supported by classical goal-setting theory, which postulates better group performance resulting from collaborative goal-setting [39]. Schnabel and Pizka [53] emphasize collaborative goal-setting in their process as well.

ID	Description	Based on	Explanation
P1	*The scaling level at which the actor operates influences their reporting responsibility and the type of necessary reporting*	R2, R7, R10, R13	*AM4, AM5* state higher-level report is product-oriented; *AM1, AM5* state team-level report is oriented on team needs and working-process; *AM5* states product-oriented report on team-level lacks overall context
P2	The recipient of the report influences what should be reported and by whom	R1, R2, R4	*PO3, AM4, AM5* state report is oriented on recipient needs; *AM5* states report has to provide needed info to plan next iteration on all levels
P3	*A hierarchy of goals should be established to determine how a goal affects the organization based on its scaling level*	R1	Lower-level goals should always clearly state to which goals on the next higher level they contribute; however, goals only relevant up to a specific scaling level might not contribute to higher-level goals, but still not be a top-level goal; Berntzen et al. [8] make a similar observation
P4	Automation of reporting becomes more important the bigger the size of the program	R14	*STE1* states with increasing size automation becomes more important to cope with increasing complexity; Murphy and Cormican report a similar relation for software measurement [45]
P5	Regulations (e.g., from government or the organization) constrain which reporting practices can be applied	R11, R12	*AM4* states workers council prohibits reporting on individual level; *PO1, PO2* state (parts of) higher-level report is standardized by top management; GMP is standardized for whole organization
P6	*Process-oriented reporting should track trends across longer periods of time to improve ability to make reliable predictions*	R4, R8	Single measurements of a metric do only show the current state; trends have to be tracked to identify changes to work processes over time and to be able to provide reliable predictions (*AM5*); In line with a continuous understanding of measurements postulated by Murphy and Cormican [45]
P7	*Team-mood should be tracked regularly using metrics*	R10	Mood of the team should be regularly evaluated in retrospectives using a metric (e.g., 5 star rating (*AM3*)); the mood-development should be tracked over time; good working processes are crucial on team-level (*AM5*)

Table 5.1.: List of all general propositions of the theoretical model (large-scale and / or agile specific propositions in italics)

5.2.2. Activities and Technologies

To carry out activities on the software system actors need to apply certain technologies [58]. According to the scope of our model, we structure the activities into the goal-setting process and the reporting process. To carry out the goal-setting process, actors need to apply techniques for definition, documentation, and communication of goals. And to carry out the reporting process, actors need to apply techniques for assembly, documentation, and communication of reports. We structure these key activities of goal-setting and reporting based on the case study. The modeling approach is influenced by Karhapää et al. [29], who modeled activities and practices for quality requirements management in a similar way in their theory. Figure 5.2 gives an overview of the activities and technologies that are part of the model.



(a) Activities (b) Technologies

Figure 5.2.: Overview of the (a) activities and (b) technologies in the model

Goal-Setting

For the goal-setting process actors need to define and prioritize the goals that should be pursued. Goal-setting responsibility is shared among all the actors of our model and we emphasize collaborative goal-setting, as discussed in the previous section. Our findings indicate that goals may originate from any internal or external stakeholders of the agile program, such as customers, upper management, Agile Teams etc. To ensure all goals from these various sources are properly defined, goal definition practices should be agreed on and applied. Only clearly defined goals should be considered by the program. The program should establish a Definition of Ready (DoR) (cf. [49]) for goals, similar to the Definition of Done for Backlog Items used in Scrum [56]. We suggest the Definition of Ready to describe the practices that actors have to apply to define a goal. In general, goal definition practices are used by actors to describe the goal and what is to be achieved. They

also describe how goal progress is to be evaluated and what is considered goal achievement. In particular, the actor should define metrics that describe how progress towards the goal is measured, and define sources for the data needed to calculate these metrics (cf. Goal Question Metric approach [2, 3]). This suggestion is influenced by Murphy and Cormican, who make a similar recommendation for the software measurement domain [45]. We identified several goal definition practices in the case study. The SMART technique is used by programs *B2, C1,* and *D1*. Objectives and Key Results were described to us by interviewees *LM1* and *DEV1* and are supported by tools throughout the whole case organization. Interviewee *AM5* described usage of program press releases to us. In program *D1*, Product Owners write artificial press releases that describe the situation that is expected once a specific goal was already achieved. Based on these findings, we propose **M2**: *Goal definition practices ensure clear understanding of goals and understanding of what to report to illustrate goal progress*. By enforcing and applying such goal definition practices in the goal-setting process, actors can avoid unclear goals (C5) and clearly define what has to be reported to illustrate progress towards a specific goal, especially on higher levels (C13). By directly linking metrics to each goal, it becomes easier for the implementing Agile Teams and other actors to evaluate the current state as well as progress towards the target state (C6). The linked metric also ensures clarity on which data has to be collected to evaluate the indicator.

Besides proper definition, goals also have to be documented. The aim is to document and store what is to be achieved, and to make this knowledge retrievable for discussion and future reference. Our model emphasizes transparency for the documentation of goals, because transparency is a key value of agile methodologies [56] and was emphasized as an important action against several challenges in the case study (*LM1, AM1, AM2, AM5*). Only artifacts documented transparently can be used for transparent decisions and goals [56]. We identified multiple possible practices for documentation of goals in literature and our case study. A very common one is the Backlog. It is used by all programs that are part of the initial case study. Backlogs are also used by several scaling agile frameworks (e.g., [25, 34, 55]). Different types of goals are stored and maintained in Backlogs. In the case study we observe that typically product, architectural, compliance, and operational goals are documented in Backlogs. However, we also observed goals of other types to be documented in Backlogs. Program *C2* is using a dedicated Backlog for Organizational Development Goals, while program *A1* is documenting all types of goals in Backlogs. Based on these findings we propose **M13**: *All goals should be maintained in Backlogs to facilitate clear understanding and transparency*. Another documentation practice we identified — which is also used in conjunction with Backlogs — is to establish a linked chain of sub-goals. This is a finding from the case study (see Section 4.4). Hence, we propose **M3**: *A linked chain of sub-goals across scaling levels should be established to facilitate transparency*. By clearly linking each goal and Backlog Item to the goals on the next higher scaling level, reporting can later show how work contributes to overall progress (C14). It gives a clear link between high-level goals and their operationalization (C10) via Backlog Items, and thus reduces arbitrariness of reporting higher-level progress (C12). As a result, the linked chain

facilitates transparency. Berntzen et al. [8] also observe that breaking down goals into a hierarchy may facilitate coordination. Another common goal documentation practice is the GMP Sheet, which was already mentioned in Section 4.3. It is based on PowerPoint slide templates. Being part of the Goal Management Process, the GMP Sheet is mandatory to use for all Domain Owners. The corporate Wiki, which is based on Confluence at the case organization, also serves for goal documentation in several programs (*B2, B3, C1, C2, D1, D2*). A benefit to public documentation of goals in the corporate Wiki is transparency, as mentioned by interviewee *AM2*. Consequently, we propose **M4**: *Goals should be documented publicly for all actors and stakeholders to facilitate transparency*. Ensuring transparency of goals by using these documentation practices addresses the challenge of unclear goals (C5). Further, having a clear documentation of all goals improves actors' and stakeholders' understanding of why certain measurements are necessary. This is demanded by Murphy and Cormican for software measurements [45], and we think should also apply for organizational measurements.

ID	Description	Addresses	Explanation
M1	*Goal-setting responsibility should be shared among actors, to facilitate collaborative goal-setting practices*	C5, C7	All actors and stakeholders can define goals of any kind (but not prioritize them); Based on statements by *LM1, AM5*; Literature finds shared, collaborative goal-setting facilitates group performance [38] and coordination in LSAD [8, 42, 53]
M2	Goal definition practices (e.g., SMART, OKRs, GQM) ensure clear understanding of goals and understanding of what to report to illustrate goal progress	C5, C6, C13	SMART is used in programs *B1, B2, C1, D1*; OKRs is used by programs *B2, C1, C2, D2*; GQM recommends directly linking metrics to goals, that answer specific questions [2]; The used goal definition technique should be documented in the Definition of Ready document
M3	*A linked chain of sub-goals across scaling levels should be established to facilitate transparency*	C10, C12, C14	Link team goals to program goals, and program goals to portfolio goals; Commonly implemented with linked Backlog Items, using Stories (team), Epics (program), Sagas (portfolio); Jira or similar solutions are used by programs *A1, B2, C1, C2, D1*; Berntzen et al. [8] make a similar observation
M4	Goals should be documented publicly for all actors and stakeholders to facilitate transparency	C5	Usage of public documentation, e.g., Wiki-pages, allows everyone to access the most recent goals (*AM2, AM3, AM5, PO3, DEV1, LM1, STE1*); Ensures transparency and mitigates unclear goals; Helps to understand rationale behind measurements, as demanded by Murphy and Cormican [45]
M5	*Goals from external stakeholders should be broken top-down along the Product Owner hierarchy to ensure consideration of dependencies and coordination*	C1, C2	Breaking down goals for the next lower level should be done collaboratively with Product Owners from both levels; e.g., via regular workshops according to development cycles similar to GMP process at case org.; Influenced by findings on importance of POs for coordination by Berntzen et al. [8]

M6	*Definition, prioritization, and communication of middle- to lower-level goals should involve Agile Teams, to ensure consideration of technical aspects and acceptance of goals by the teams*	C2, C4, C7	Agile Teams should always be involved in the 'how' of goals (*AM2*); E.g., via participation in workshops for breaking down goals in addition to participation in Refinements and Plannings on team-level; Murphy and Cormican also suggest to involve developers [45]
M7	*Process-oriented reporting should be fully automated using Backlogs (e.g., using tools like Jira and Dashboards)*	C15, C16, C18	Process-oriented reporting is conducted continuously, thus effort has to be minimized using automation; Automation can be facilitated by using quantitative reporting, e.g., via Jira Dashboards, Value Stream Dashboards (*PO1, AM2, AM3, AM4, AM5, LM1, STE1*)
M8	*Product-oriented reporting should be partially automated by linking goals to higher-level goals and predefine metrics for each goal, based on the linked subgoals*	C15, C18	Predefined metrics for product-oriented reporting should be calculated automatically as far as possible; Manual, qualitative evaluation should still be added by the reporter (*STE1*); Korpivaara et al. suggest in their study that metrics should be connected to goals [31], and Murphy and Cormican even suggest to link *every* metric to a goal [45]; Also helps to show the impact and relevance of metrics, as suggested by Oza and Korkola [46]
M9	*Focus of team-level reporting should be on artifacts that were produced in the last iteration (development-oriented reporting) and on overall process monitoring, rather than on individual work*	C3, R7, R10	Focusing reports on individuals is detrimental to group performance [38]; Reports focused on artifacts are dominant on team-level [33]
M10	Compensation of actors should not be linked to reporting outcomes	C17	Conflicts of interest, e.g., compensation based on reports, should be avoided (C17); Otherwise, such conflicts may cause manipulation (cf. C17, [45, 48]); Reports have to be used in regular retrospectives to derive improvement actions
M11	An arena should be established for teams to explicitly report on work that does not directly contribute to the most important goals ("routine work")	C8	Work that is not directly contributing to goals has to be made visible and be appreciated; Otherwise people will avoid such tasks (*LM1, AM3*)
M12	Focus of product- and development-oriented reporting should not be on metrics only	C8	Except for process-oriented reporting, reports should not only focus on metrics; qualitative evaluation and explanations are important to transfer information (*AM5*)
M13	All goals should be maintained in Backlogs to facilitate clear understanding and transparency	C5	Goals of all types are documented and maintained in Backlogs by programs *A1, B1, B2, B3, C1, C2, D1*, and *D2*.

Table 5.2.: List of all mitigation propositions of the theoretical model, which address one or more identified challenges (large-scale and / or agile specific propositions in italics)

Finally, defined and documented goals also have to be communicated among the different actors in the model. Because in large-scale settings not all actors can be in direct contact with the customer, documenting and communicating goals is key to ensure shared direction [42]. Since our approach emphasizes collaborative goal-setting this is of particular relevance. Four interviewees in the case study (*PO1, LM1, AM1, AM5*) emphasized the importance of communication for goal-setting. For communication, the cadence and scaling level attributes of the goal-setting process are relevant. A balance has to be achieved between the challenges of external dependencies that limit autonomous goal-setting of the Agile Teams (C1) and missing attachment of Agile Teams to goals not set by themselves (C7) [8]. While involvement of teams in defining the goals is central to agile software development, Agile Teams cannot be involved in all levels of goal-setting across the program or organization [8, 42]. Instead, we suggest to establish a hierarchy of goals. In our approach we propose **P3**: *A hierarchy of goals should be established to determine how a goal affects the organization based on its scaling level*. This recommendation is based on our findings in the case study (*STE1, AM1, BE1, AM2, LM1, AM4, AM5, PO2, PO3*) and supported by literature [8]. Goals of the Portfolio should ideally be reached by achieving sub-goals at the program-level, which in turn can be finished by realizing their sub-goals for teams. As a consequence, the scaling level at which a goal affects the organization is important to determine how the goal fits into this hierarchy and which actors drive communication. Schnabel and Pizka [53] use a similar concept they call "vertical distribution" of goals. However, their concept does not consider a hierarchy across multiple organizational scaling levels. Based on the proposed hierarchy of goals, we further propose **M5**: *Goals from external stakeholders should be broken top-down along the Product Owner hierarchy to ensure consideration of dependencies and coordination*. This addresses the challenge of external dependencies limiting autonomous goal-setting of teams (C1). The proposition is influenced by observations on the importance of Product Owners for coordination in large-scale agile development by Berntzen et al. [8]. Using the goals broken down from top, teams can autonomously define their own goals while relying on higher levels to consider dependencies. Additionally, we propose **M6**: *Definition, prioritization, and communication of middle- to lower-level goals should involve Agile Teams, to ensure consideration of technical aspects and acceptance of goals by the teams*. This addresses the challenge of missing attachment of teams to goals (C7). Murphy and Cormican also suggest that teams should be involved in defining appropriate metrics [45]. The combination of propositions **M5** and **M6** is similar to the concept of "top-down thinking and bottom-up acting" by Schnabel and Pizka [53]. However, Schnabel and Pizka only consider stakeholders ("top") and developers ("bottom") [53], while in our approach we consider multiple organizational scaling levels that are present in large-scale development, and several other actors and stakeholders specific to large-scale agile methodologies. Further, in contrast to Schnabel and Pizka, we also emphasize that Agile Teams in large-scale agile development should not only act upon given goals ("bottom-up acting"), but should be actively involved in the definition of goals as well ("thinking").

Reporting

For the reporting process we identified three types of key practices that actors should apply: assembly, documentation, and communication of the report. As explained in the description of the actors, our approach differentiates between process-, product-, and development-oriented reporting, based on Stettina and Schoemaker [59]. Assembling reports is the area where most of the identified challenges from the case study are located. In general, assembling of reports should be automated as far as possible (*PO1, AM4*). Based on our observation that especially large-scale programs rely on automation (R14), we propose **P4**: *Automation of reporting becomes more important the bigger the size of the program*. Thus, in large-scale agile environments automation is of high importance. This proposition is supported by Murphy and Cormican, who observe a similar relation for software measurements [45]. To mitigate challenges of missing automation (C15) and gathering data (C16), and to improve the cost-benefit trade-off of regular reporting (C18), our approach emphasizes automation of report assembly to the highest possible extent. We propose **M7**: *Process-oriented reporting should be fully automated using Backlogs*. Fully automated process-oriented reporting enables continuous reporting with low effort. Based on our findings in the case study and in literature, the process-oriented report on all scaling levels should mostly be using Backlog analysis with metrics. All metrics identified in the case study are listed in Section 4.4. Our observations further indicate, that process metrics should be tracked over time to uncover trends and improve the planning capability for upcoming iteration (R4). Thus, we propose **P6**: *Process-oriented reporting should track trends across longer periods of time to improve ability to make reliable predictions*. This proposition is in line with the continuous understanding of measurements postulated by Murphy and Cormican [45]. In particular, interviewees *AM3* and *AM5* emphasized the importance of tracking team-mood over time to be able to correlate mood trends with trends of other metrics. We propose **P7**: *Team-mood should be tracked regularly using metrics*.

Product-oriented reporting, on the other hand, is typically less automated. This can be mitigated by proper goal-setting as described above (especially proposition **M2**). Thus, we also propose **M8**: *Product-oriented reporting should be partially automated by linking goals to higher-level goals and predefine metrics for each goal, based on the linked sub-goals*. Defining metrics for each goal, combined with a target value for the metric, and ensuring that sub-goals are properly linked to their parent goals in the Backlog facilitates automated evaluation of product progress (C15) (cf. [45]). Development-oriented reporting, which is based on the actual artifacts produced in an iteration and presenting them to stakeholders [33], hardly can be automated. Based on our observations in the case study to avoid putting external pressure on Agile Teams (R7) and that the working process is most important on team-level (R10), we further propose **M9**: *Focus of team-level reporting should be on artifacts that were produced* (i.e., development-oriented reporting) *and overall process monitoring* (i.e., process-oriented reporting), *rather than on individual work*. Focus should be on produced artifacts (cf. [33]), how well the team is functioning, and ensuring they can deliver what they set out to achieve (*AM5*). According to goal-setting theory, focusing on individual

work could even be detrimental to team performance [38]. Thus, as a side-effect, this focus on process-oriented reporting addresses the challenge of not being allowed to report on individual employees (C3). Reporting, as found in the case study, is also influenced by regulations from government and organizational regulations (R11, R12). Based on this observation, we also make the general proposition **P5**: *Regulations (e.g., from government or the organization) constrain which reporting practices can be applied*.

Interviewees *PO1* and *BE1* described situations of conflict of interest to us, if the actor that assembles a report is compensated based on said report (C17). To mitigate this challenge of conflicts of interest (C17), we propose **M10**: *Compensation of actors should not be linked to reporting outcomes*. Based on agile values (cf. agile manifesto [6]), reports should not be seen as ways to formally control people but rather as a means for continuous improvement. Murphy and Cormican similarly emphasize that software measurements and reports should be seen as a means for continuous improvement to avoid fear of exposition and data manipulation [45].

Once all the data and information necessary for assembling the report has been collected, it should be documented. The same principles of goal documentation also apply here. Our approach emphasizes transparency for all actors, to allow everyone to access reports and retrieve information as needed. Documentation practices we identified include the GMP Sheet, public pages in the corporate Wiki, and presentation slides on a publicly shared folder (see Section 4.4).

Communication of reports is the third area of practices that actors have to apply. Since reports are assembled to consolidate data and inform other actors and stakeholders, communicating the report is a central aspect of the reporting-process. As proposed above (propositions **P1**, **P2**), the correct type of reporting depends on both the scaling level of the actor and on the stakeholders that the actor is reporting to (R1, R4). While reports should be oriented around the goals they are addressing and the recipients to whom is reported, we identified the challenge that routine work is often underrepresented in reports and consequently remains unappreciated (C8). To mitigate this challenge, in our approach we propose **M11**: *An arena should be established for teams to explicitly report on work that does not directly contribute to the most important goals ("routine work")*. We emphasize to also report on work that does not directly contribute to the most important goals.

In general, we observe that interviewees try to keep a balance between quantitative and qualitative reporting. As process-oriented reporting is primarily based on quantitative metrics in our approach, we recommend to also incorporate qualitative parts into product-oriented reporting, and to primarily focus on artifacts in development-oriented reporting, as described by Settina and Schoemaker [59]. We formalize this in proposition **M12**: *Focus of product- and development-oriented reporting should not be on metrics only*.

In Figure 5.3 all propositions are shown and linked to the respective entities in the theoretical model, using the UML-like notation proposed by Sjøberg et al. [58].

While we defined propositions to address most of the identified challenges, for some challenges we did not define any propositions to address them. These challenges are C9, C11 and C19. To address challenge C9: *External contracts limiting Feature Team working model*

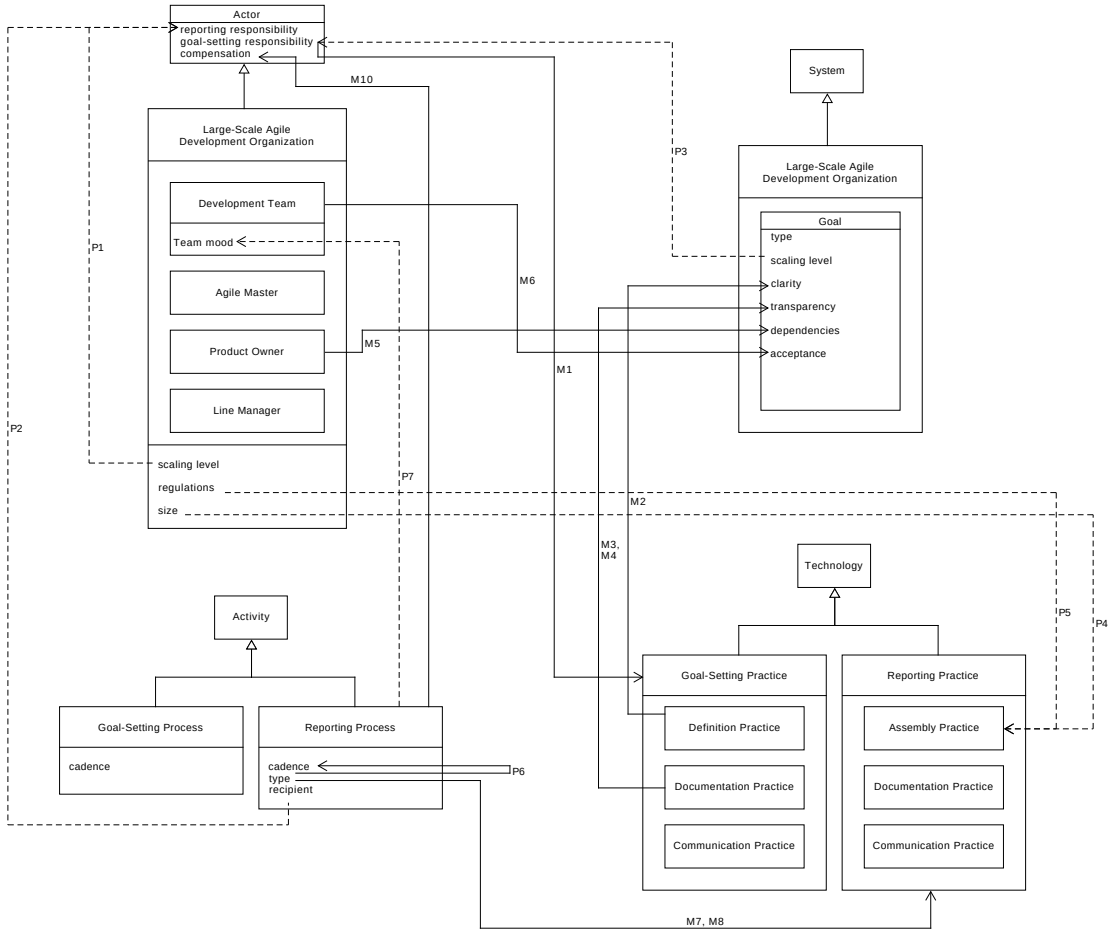


Figure 5.3.: The theoretical model that formalizes the basis for our goal-setting and reporting approach. All constructs and depictable propositions are shown. General propositions are depicted with dotted arrows, mitigation propositions with solid arrows. Visualized using the UML-like notation by Sjøberg et al. [58].

it would be required to change German labor law because it is currently only allowed to assign work to external teams via a "bridge" contact person. This would be clearly out of scope for organizational goal-setting and especially for this master's thesis. For challenge C11: *Resource constraints for goal-setting and reporting* the operating mode of the organization itself is the cause for the challenge. The organization itself could resolve this challenge by reallocating resources. However, in current market situations this seems not to be possible, hence this challenge inevitably will persist. Finally, to address challenge C19: *Reporting delayed due to hierarchy* more direct communication seems to be the obvious solution. However, in large organizations it is simply not possible for every employee

to directly communicate with every other employee — indirection in communication is inherent to large organizations.

5.3. Process Model

Based on the formalized constructs and propositions, in this section we describe a process model for goal-setting and reporting in large-scale agile software development. It depicts the overall picture and describes how the individual propositions fit together. The goal-setting part of our process is influenced by Schnabel and Pizka [53], and the reporting part of our process is influenced by the ISO/IEC 15939:2007 process [24].

5.3.1. Goal-Setting Process

Figure 5.4 shows the key activities of the process for goal-setting. The individual steps of the model are explained in detail in the following sub-sections.

Identify goal

In the first step, internal actors and external stakeholders of the large-scale agile development environment identify goals based on their needs. External stakeholders are stakeholders that are not directly involved in the actual agile working environment, but have an interest in the developed portfolio or product. These stakeholders can be from the same organization (e.g., the Board of Directors) or from outside of the organization (e.g., customers). Needs of actors are considered *internal*, while needs of stakeholders are considered *external*. This consideration of goals from different actors and stakeholders in the first steps of the process realizes the proposition of shared goal-setting responsibility (M1).

Define and document goal

Each identified goal has to be defined and documented according to the definition and documentation practices of the portfolio or program. To realize proposition M13, Backlog Items that contribute to a goal should be directly linked to this goal, using a dedicated Backlog tool, e.g., Jira. For clear differentiation, on each organizational scaling level a different type of Backlog Item should be used to document Backlog Items that contribute to goals. In Jira this can be Saga on portfolio-level, Epic on program-level, and User Story on team-level. Each goal, in turn, has to be linked to the next higher-level goals that it is contributing to (P3, M3). Further, we recommend to formulate quantifiable Key Results for each goal on portfolio- and program-level, similar to the Goal Question Metric Approach (GQM) [2]. Each Key Result should be calculated based on one metric. This metric, in turn, should be calculated based on the sub-goals or Backlog Items that contribute to the goal. Using this two-fold approach ensures a traceable hierarchy of goals from top to bottom, makes each goal quantifiable, and transparently documents how each of the sub-goals and

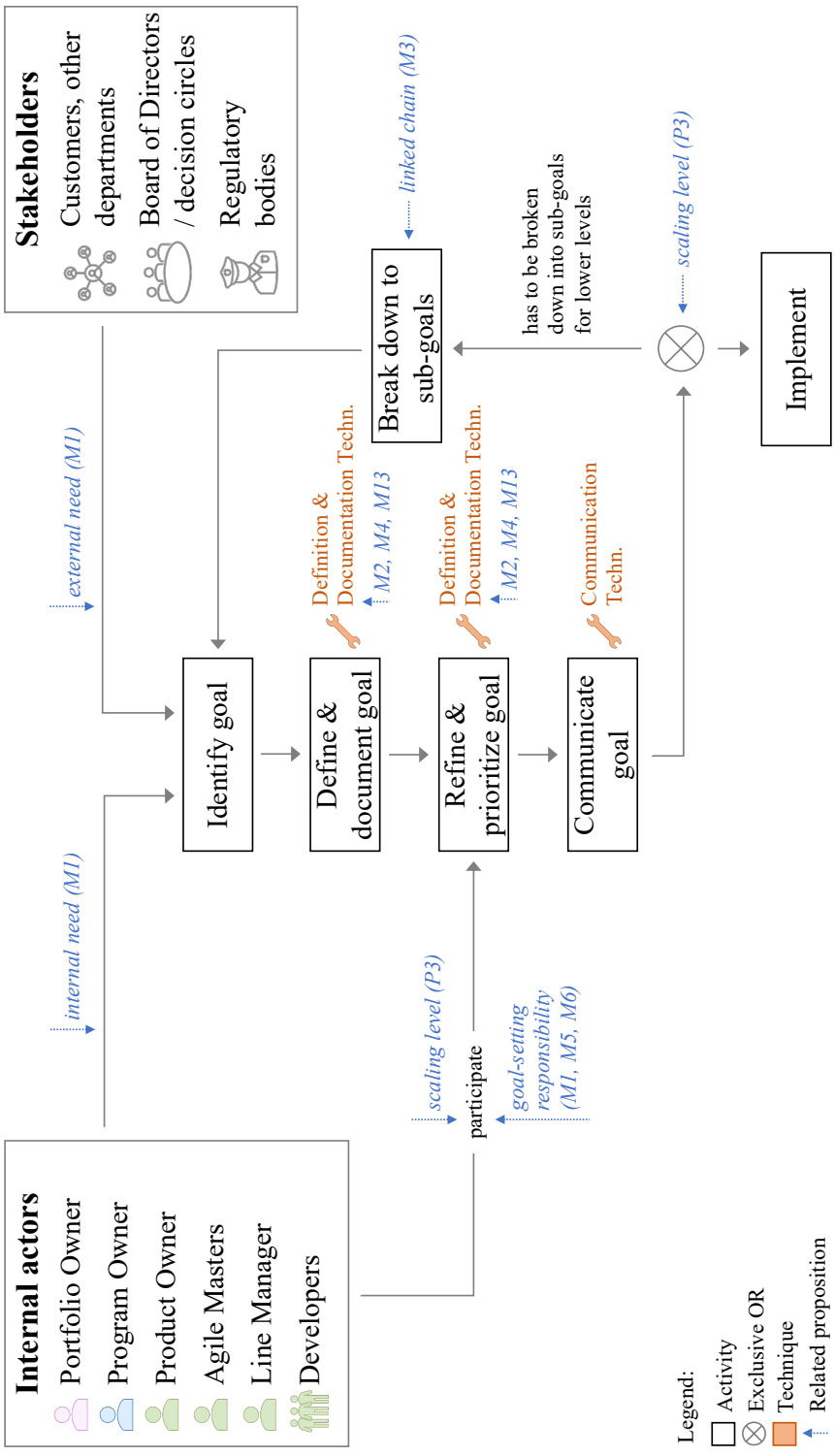


Figure 5.4.: Visualization of the recommended goal-setting process

Backlog Items contributes to overall progress. It thus realizes propositions P3, M2, M3, and M4. In cases where the Key Result cannot be based on a sensible metric, we recommend to use the ratio of linked sub-goals or Backlog Items that are already finished as an indicator. In such a case, the goal should specify how the progress can be reported qualitatively. Our recommended structure of goals using linked Backlog Items and Key Results is visualized in Figure 5.5. Using such a structure realizes the proposition to establish a linked hierarchy of goals and Backlog Items (M3). Depending on the scope and level of impact of a goal, Backlog Items of the corresponding organizational level should be linked. This realizes the proposition that the scaling level determines how a goal fits into the hierarchy of goals (P3).

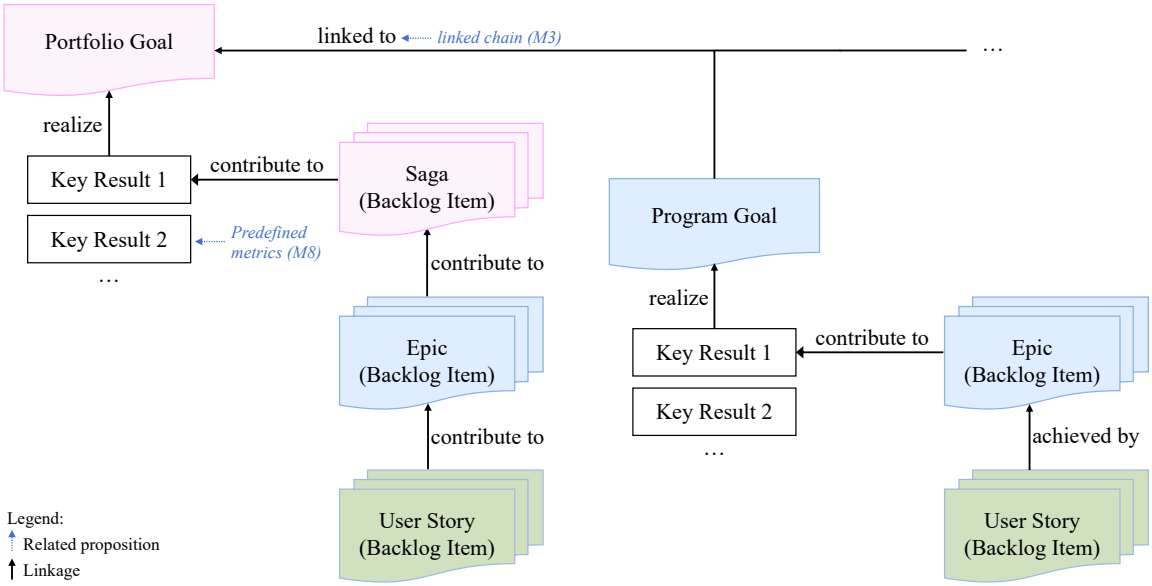


Figure 5.5.: Suggested structure and linkage of goals using a combination of Backlog Items and Key Results

Refine and prioritize goal

Based on the iteration cadence on the scaling level at which the identified goal is relevant, it has to be refined and prioritized by the Portfolio / Program / Product Owner and, on team- and program-level, the Agile Team. To realize M5, goals coming from the head of the department, the Board of Directors, or other departments of the organization should be introduced to the program via the Portfolio Owner.

Refinement and prioritization should be conducted as recommended by the chosen scaling agile framework, e.g., SAFe or LeSS. The refinement and prioritization should be conducted collaboratively, because goal-setting responsibility is shared among all actors. We

recommend to conduct a refinement and prioritization meeting at the beginning of each new iteration at each scaling level. All actors that are active on this scaling level should participate in this meeting, as well as the Product Owner(s) of the next lower scaling level. This allows the goals to be broken down along the Product Owner hierarchy, and ensures consideration of potential dependencies. This design decision is influenced by findings from literature, which indicate that frequent communication between Product Owners is important for coordination in large-scale agile environments [8]. At team-level and, whenever possible, on program-level Agile Teams should participate in the refinement and prioritization meetings, to realize M6. The teams are responsible for consideration of technical feasibility and solution strategies for the goals and derived requirements. We recommend this combined approach of top-down definition of goals and bottom-up inclusion of Agile Teams to ensure both consideration of dependencies to other programs or teams and (technical) solution strategies. Further, shared responsibility for goal-definition and prioritization makes sure all actors commit to the goals. This realizes propositions M1, M2, M5, and M6. To ensures attachment of Agile Teams to the goals, the teams have to be involved in the goal-setting process at least on team-level and preferably also on program-level. In large-scale agile development it is crucial for all teams to understand the goals and to find common ground for what is to be achieved [42]. Otherwise, teams may perceive goals as requirements and deliverables only [42], causing them to develop own goals incompatible with program goals. This can result in decrease of team performance [39].

Communicate goal

Once a goal has been refined and prioritized by the actors of the respective scaling level, the commitment has to be communicated to all stakeholders and actors. At the very least, we recommend to grant all stakeholders and actors access to the Backlogs at all scaling levels.

Break down to sub-goals

As a result of our recommendation to break down goals into sub-goals, the goal-setting process has to be iterative. Breaking down goals causes the need for a new cycle of defining, documenting, and communicating the sub-goals at the next lower scaling level. Breaking down goals into sub-goals is a non-trivial task, as our observations have shown. Hence, we recommend the new cycle of defining the sub-goals again to start with the first step in the goal-setting process, *identify goal*, and then run through the whole process. As mentioned above, Figure 5.5 shows our recommended approach for goal- and Backlog Item-linkage. The need for further breaking down a goal typically depends on the scaling level of the goal. Goals on portfolio- and program-level should be broken down into sub-goals to ensure work of Agile Teams is connected to the higher-level goals. Goals on team-level do not need to be broken further down. Instead, goals on team-level are implemented by the Agile Teams via according Backlog Items (e.g., User Stories). Higher-level goals,

in turn, are implemented by implementing their respective sub-goals and linked Backlog Items on that level. Ideally, every goal and requirement the Agile Teams are working on is linked to a higher-level goal.

While the steps *identification*, *prioritization*, and *implementation* are also present in the process by Schnabel and Pizka [53], we add the steps *definition & documentation* and *communication*. We make this addition to put emphasis on collaborative goal-setting and transparency for all actors and stakeholders. Further, while *vertical distribution* is a simple process-step in the approach by Schnabel and Pizka [53], in our approach goals are iteratively broken down into sub-goals and thus vertically distributed "by design". This is due to the explicit consideration of large-scale environments in our approach, whereas Schnabel and Pizka [53] do not consider multiple scaling levels.

5.3.2. Reporting Process

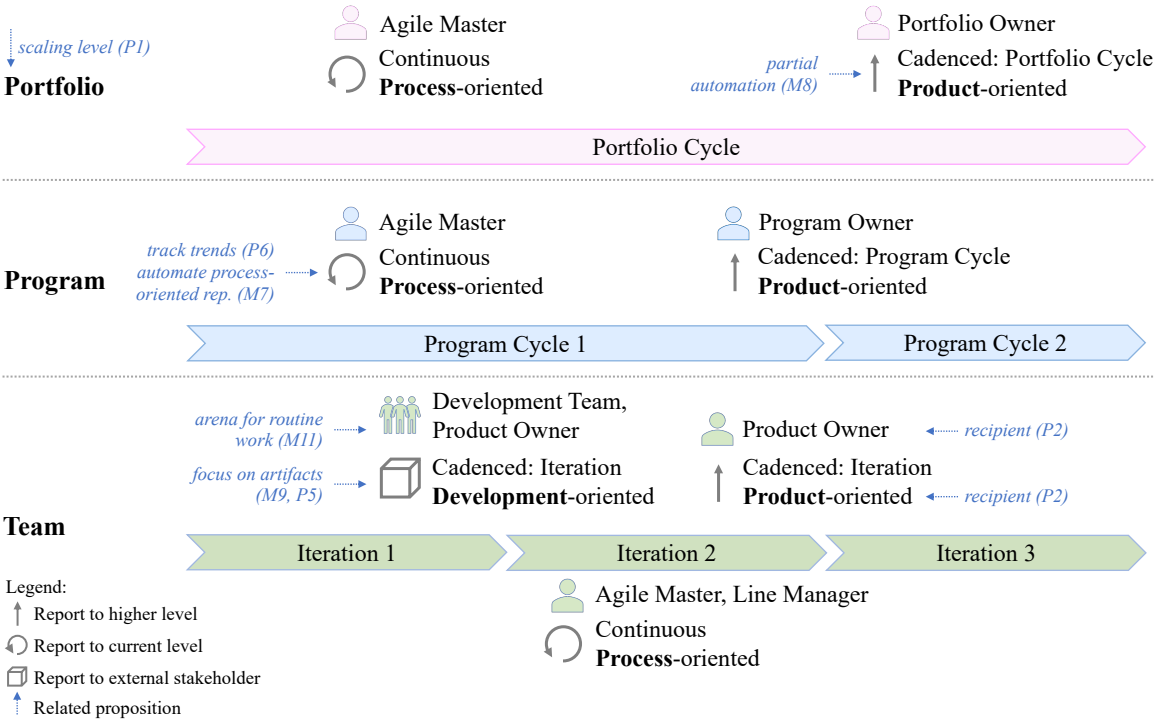


Figure 5.6.: Visualization of the different types of reporting on each scaling level based on the observations from the case study, as well as relevant propositions

For the reporting process in our model, we differentiate between the three types of reporting *development-oriented*, *process-oriented*, and *product-oriented*. This differentiation is taken from existing literature [59]. However, because our approach is focused on large-scale environments, in addition to extant literature we also consider the different scaling

levels (*team*, *program*, *portfolio*) at which reporting takes place. This considers the proposition that the scaling level of an actor influences the reporting (P1). Further, our process suggests to connect the reportings across the different scaling levels and details when each reporting should be conducted. Figure 5.6 visualizes the presence of the three types of reporting on the different scaling levels, based on our observations in the case study (see Section 4.4), with the addition of indicators for relevant propositions.

Product-oriented reporting

Product-oriented reporting should be present on all three scaling levels. In line with literature, it is the responsibility of the Product / Program / Portfolio Owner in our approach [59]. Product-oriented reporting should appear in a rhythm, aligned with the iteration cadence on the respective scaling level. To realize proposition M8, it should be at least partially automated to reduce efforts, using automatically calculated metrics defined in the goals. Figure 5.7 shows example metrics that can be used in product-oriented reporting. As we observed, typically at least a Say-Do-Rate should be calculated on each level, to keep an overview of how much of the committed work has been achieved. However, based on our observations, product-oriented reporting typically contains also a qualitative part to give reporters (i.e., the Product Owner hierarchy) the opportunity to explain and communicate information besides metrics. This is based on proposition M12. Thus, product-oriented reporting is a mixture of quantitative and qualitative reporting. As of propositions M3 and M8, focus of product-reporting is on progress towards the goals that contribute to the goals on the next higher-level. E.g., the Product Owner should report progress on the ten User Stories that contribute to a Key Result of a quarterly Epic on program-level. This can be done using a Say-Do-Rate to show how many of the ten committed User Stories have actually been finished in the last Sprint. On program-level, in turn, the Program Owner can report progress of an Epic to portfolio-level by calculating the metrics that are defined in the Key Results of the Epic. Often, product-oriented reporting also needs qualitative evaluation to inform about unforeseen changes or impediments (as formalized in proposition M12). In such cases we suggest to at least add a color coded explanation to the affected metric(s), using orange for impediments that can be solved autonomously and red for escalation (similarly as observed in the case study).

Process-oriented reporting

Process-oriented reporting should also be present on all three scaling levels. It is the responsibility of the Agile Master at the respective scaling level [59]. Focus is on work processes [59]. To implement propositions M7 and P6, our case study findings indicate this reporting should be fully automated to enable it to continuously monitor changes and trends in working processes. We suggest automation using Dashboards as offered by tools, e.g., Jira or eazyBI. Hence, it is a quantitative reporting that exclusively uses metrics. Figure 5.7 shows example metrics that can be used in process-oriented reporting. Typi-

cal process-oriented metrics are the average cycle and lead time at each scaling level. On program- and team-level Velocity trends (see Velocity Forecast in Appendix B.2 for detailed explanation) are important to be able to reliably plan upcoming iterations. Agile Masters should track the chosen metrics over longer periods of time, to be able to identify trends or changes over time. As of proposition P2, the identified trends and changes are reported to internal actors (primarily teams and Product Owners) to facilitate planning. Suggested metrics to monitor in process-oriented reporting include average lead and cycle time, or the ticket resolution ratio. Additionally, Agile Masters should keep track of team-mood or happiness metrics (e.g., simple five-star ratings via small surveys in chat apps) over time, as proposed in P7. While not directly being a process-measurement, it should be tracked continuously as part of the process-oriented reporting. This allows to compare and correlate changes in process-metrics with team-mood.

Development-oriented reporting

Finally, as proposed in M9, development-oriented reporting should be present on team-level. This type of reporting is primarily the responsibility of the development team [59], supported by the Product Owner as we observed. Development-oriented reporting should be conducted at the end of each team iteration [59], and in cooperation with the other teams, at the end of each program cycle. Unlike the other two types of reporting, development-oriented reporting focuses on the artifacts created during the past iteration (i.e., working software) [59], not the artifacts that represent the goals (i.e., Backlog Items). The reporting typically is done in the review or demo meeting [59], depending on the used scaling agile framework. Based on our observations the recipients are customers and users, and given the focus on artifacts, the report is of qualitative nature. However, we suggest to collect quantitative data during the report meetings, to be able to calculate metrics such as user satisfaction, test duration, or team-mood after the development reporting took place. This data can be used as input for process-oriented reports to detect trends across multiple iterations. Figure 5.7 shows example metrics that can be collected in development-oriented reporting, to allow for trend observation in process-oriented reporting. These typically include software measurements such as test coverage, test success rate, number of deployments, but also customer happiness (e.g., using a simple rating from the customer) and technical debt ratio (the number of closed technical debt items vs. newly created ones in the last iteration). Apart from this, development-oriented reporting should also report on work done in the teams that does not directly contribute to specific goals, as proposed in M11. We suggest to assign a slot during iteration review meetings, dedicated for review of such routine and maintenance work. This is essential to keep up motivation of the teams to work on such tasks.

		Type of Reporting		
		Development	Product	Process
Scaling Level	Portfolio	-	• Saga Say-Do-Rate • Security Patch Level	• Avg. Saga Cycle Time, Lead Time • Estimated vs. Unestimated Items
	Program	• Customer Happiness • Technical Debt Ratio	• Epic Say-Do-Rate • Security Patch Level	• Program Velocity Forecast • Avg. Epic Cycle Time, Lead Time • Estimated vs. Unestimated Items
	Team	• Customer Happiness • Technical Debt Ratio • Test Line Coverage • Test Decision Coverage • Test Success Rate	• Story Say-Do-Rate	• Team Velocity Forecast • Team Mood Trend • Avg. Story Cycle Time, Lead Time • Burn-Down Chart • Cumulative Flow Diagram

Figure 5.7.: Example metrics from the case study for each type of reporting and scaling level; similar to the one Stettina and Schoemaker created for their case study [59]

All three types of reporting should follow a standard process of four activities. We suggest a process based on the propositions we formalized in the theory, and align it with our goal-setting process. Figure 5.8 visualizes the key activities of the process for reporting. The goal-setting and reporting processes are interdependent and influence each other. Especially the recommended hierarchy and linkage of goals and Backlog Items (Figure 5.5) is relevant to reporting, to realize proposition P4. It allows for automation and systematic calculation of defined Key Result metrics, using dedicated Backlog tools (e.g., Jira).

Collect data

In the first step of the suggested reporting process the necessary data has to be collected. The needed data are determined by the goals that are reported. In cases of goals that define Key Results with metrics the needed data depend on the input for the calculation of these metrics. For goals where no quantitative metrics were defined the needed data may be of qualitative nature, e.g., user feedback. To realize proposition P4, we recommend to base as many reports as possible on data that can be automatically generated and collected from the Backlogs or other dedicated systems. Using the exemplary Backlog structure displayed in Figure 5.5, the status of goals can be automatically derived from linked sub-goals via their contributing Backlog Items. For goals without defined Key Result metrics we suggest to rely on simple Say-Do-Rates for quantitative reporting. This allows to keep an overview of whether the committed sub-goals have actually been achieved while granting autonomy of the lower levels which goals to commit to.

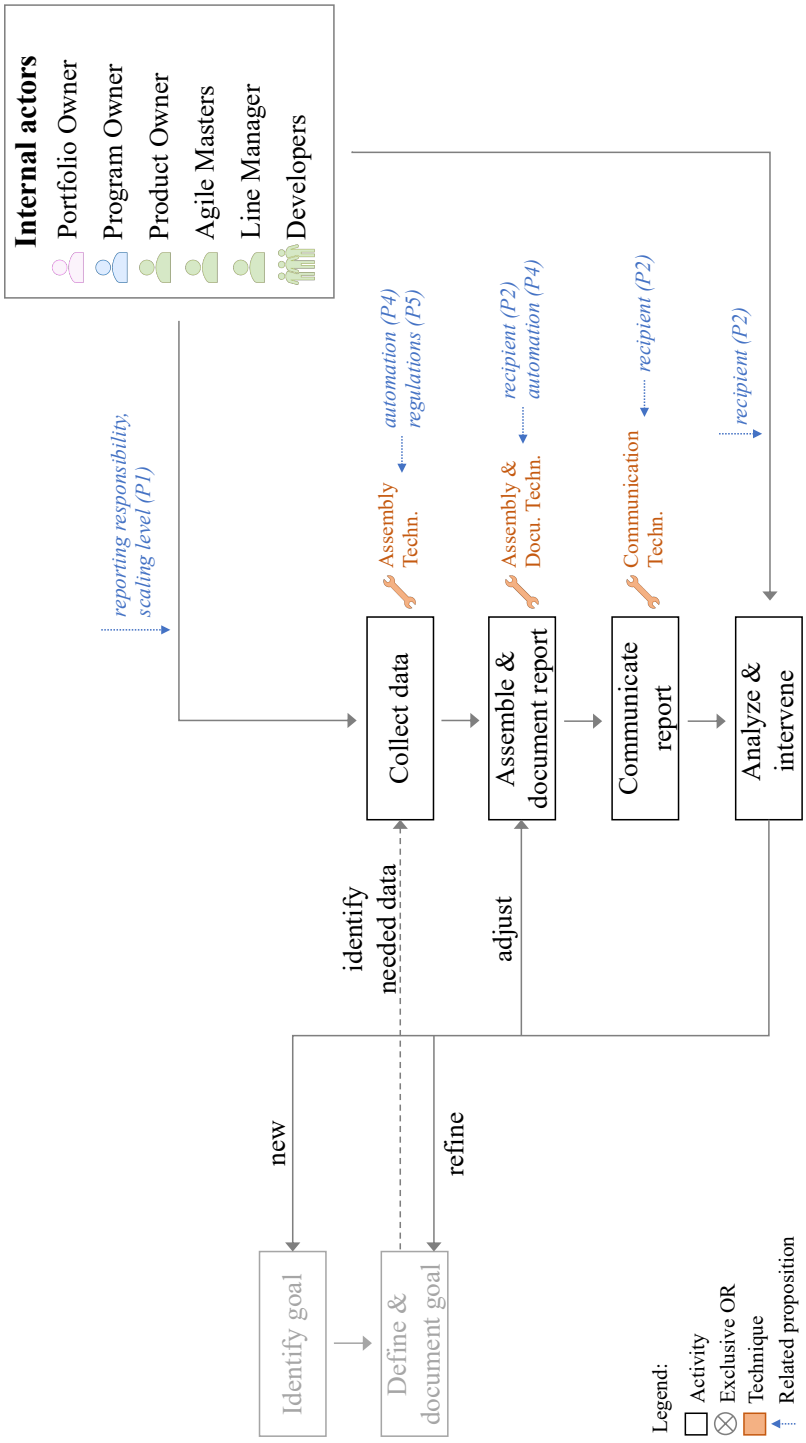


Figure 5.8.: Visualization of the recommended reporting process (greyed out steps of goal-setting process are added for reference)

Assemble and document report

Using the collected data, the report can be assembled and documented. This includes the calculation of used metrics for the goals that are reported. Depending on the scaling level and recipient different tools and forms of documentation may be used, as proposed in P1 and P2. We recommend to publicly document all reports using Dashboards and Wiki-pages to ensure transparency across actors and scaling levels (similar to proposition M4).

Communicate report

After assembly the responsible reporting actor has to actively communicate the results to the recipient. For cadenced reports (i.e., product- and development-oriented) we suggest to setup dedicated meetings or slots in existing meetings for communication and analysis of the results. Continuous reports (i.e., process-oriented) should be communicated to recipients if significant changes are observed by the Agile Master. For this we suggest ad-hoc meetings.

Analyze and intervene

Finally, depending on the analysis the recipient draws from the report, an intervention might be necessary. We define three types of interventions. First, changes might be necessary to existing goals. Thus, the existing goals need to be refined again and the changes need to be documented. The goal then has to go through the standard steps of prioritization and communication again in the next cycle. This ensures transparency for all actors and stakeholders as well as propagation of changes to other scaling levels. Second, new goals may emerge as a result of the analysis. And third, adjustments might be necessary to the way of how the report is assembled in the following cycles. This might be the case if the recipient decides that the reported information is insufficient, inconclusive, or redundant.

Parts of our reporting process are influenced by step 4.3 of the ISO/IEC 15939:2007 software engineering measurement process [24]. Both processes comprise data collection (ISO step 4.3.2) and analysis (ISO step 4.3.3), as well as communication of results (ISO step 4.3.4). However, our process makes several additions and changes that are specific to large-scale agile environments. First, we inter-connect our reporting process with the goal-setting process. The goals defined in the goal-setting process influence data collection. The results gathered from the analysis of the report in turn influence the steps *identification* and *definition and documentation* of the goal-setting process. This allows for continuous, iterative improvement of the goals and the reporting process itself, which fits the agile value of harnessing change (cf. agile manifesto [6]). Second, in our process the whole report, including the results of analysis, should be made available to all interested parties. The ISO process only suggests to communicate the results of analysis. This difference is due to the empha-

sis of our approach on transparency. Sharing the whole report, not only the results, avoids manipulation and "green-shifting" (i.e., deriving only favorable results from data) of analysis results (C12). Third, our process adds a dedicated step for *assembly and documentation* of reports. This is due to our observations that these activities constitute a considerable effort in reporting. Data needs to be processed and visualized between collection and analysis, which is of vital importance and can be done using various techniques and approaches. Fourth, our process explicitly considers reporting across organizational scaling levels and with different focus areas. This is specific to large-scale environments. Our suggested approach differentiates between product-, process-, and development-oriented reporting. Using our suggested linkage structure for goals and Backlog Items, reports become interconnected across organizational scaling levels. Higher-level reports are based on linked sub-goals and predefined metrics. Such large-scale settings are not explicitly considered by the ISO/IEC 15939:2007 software engineering measurement process.

6. Artifact Evaluation and Improvement

This chapter presents the evaluation of the propositions that we conducted at the case organization. First, Section 6.1 describes the methodology that is used to evaluate the propositions. Subsequently, the results of the evaluation are presented and discussed in Section 6.2. For every proposition, the collected feedback is summarized. In the final Section 6.3 adjustments to the propositions and process models are described, which we derive from the evaluation feedback.

6.1. Methodology

The goal of the evaluation was to investigate whether practitioners of large-scale agile development agree to the propositions that we made based on our findings from the initial case study and literature. Further, we wanted to collect qualitative feedback and insights on the propositions. We decided to focus on the evaluation of the propositions because of two reasons. First, the propositions are the central artifact to answer the third research questions, since the mitigation propositions intend to address identified challenges. Second, the propositions form the basis and central building blocks of the process models.

No.	Alias	Role	LSAD experience	Softw. Dev. experience	Duration (h:m)
1	PO1	Product Owner	3 - 5 years	11 - 15 years	0:25
2	BE1	Business Expert	6 - 10 years	6 - 10 years	0:38
3	AM2	Agile Master	3 - 5 years	6 - 10 years	0:52
4	**PO4**	Product Owner	3 - 5 years	6 - 10 years	0:41
5	**PO5**	Product Owner	6 - 10 years	11 - 15 years	0:37
6	**AM6**	Agile Master	6 - 10 years	11 - 15 years	0:37
7	**PO6**	Area Product Owner	3 - 5 years	> 20 years	0:38
8	LM1	Line Manager	6 - 10 years	6 - 10 years	0:38
9	**AM7**	Agile Master	6 - 10 years	> 20 years	0:38
10	DEV1	Developer, Software Architect	1 - 2 years	6 - 10 years	0:27
11	PO3	Product Owner	3 - 5 years	11 - 15 years	0:25

Table 6.1.: Overview of the evaluation interview participants (participants that did not take part in the initial case study interviews are outlined bold)

To achieve this objective of the evaluation, we decided to conduct a second round of semi-structured interviews with practitioners from the case organization. In general, the chosen evaluation can be described as *observational* because we are using our case organization to evaluate the artifact, and as *descriptive* because we are leveraging knowledge and insights from practitioners to collect arguments for the artifact's utility [23].

In the first section of the interviews, similar to the initial case study interviews, we asked interviewees to describe their role and experience with large-scale agile software development. To evaluate the propositions, in the second section we had interview participants answer two questions for each proposition. In the first question, we asked them to assert whether they agree or disagree to the proposition, using a five-point Likert scale [37]. Possible answers included *strongly agree* (1), *agree* (2), *neither agree nor disagree* (3), *disagree* (4), and *strongly disagree* (5).

After they had chosen their answer to the first question, we also asked the interviewee to explain why they chose this answer and give further thoughts on the proposition. We then coded and analyzed these qualitative answers similarly to the initial case study interviews. Again, we conducted a first cycle of coding and assigned descriptive codes to relevant passages of the feedback [41]. Using this inventory of topics for the second cycle coding, we then inductively grouped them using pattern codes for recurring concepts [41]. We chose this approach because it allowed us to uncover recurring topics across the different interviews, while avoiding to introduce any bias from the researchers via a pre-defined coding structure.

In total, we conducted 11 interviews for the evaluation. An overview of the participants is depicted in Table 6.1. The complete questionnaire that we used for the evaluation interviews can be found in the Appendix A.2.

6.2. Evaluation of Propositions

In this section we describe the evaluation results that we collected in the evaluation interviews. For each proposition, the statistical results are presented and the qualitative feedback from the interview participants is discussed. Figures 6.1 and 6.2 depict an overview of the evaluation results for the general propositions as well as the mitigation propositions, respectively.

P1: The scaling level at which an actor operates influences their reporting responsibility and the necessary type of reporting.

The evaluation of proposition P1 resulted in an average rating of 1.73 with a standard deviation of 0.62. Interviewees identified two major dimensions of reporting responsibility that are influenced by the scaling level of an actor. The first dimension is the focus of the reporting. Participants pointed out that higher level reporting has to aggregate information to a higher degree than lower-level reporting (*PO1, PO4, PO5, PO6, DEV1*). Examples for

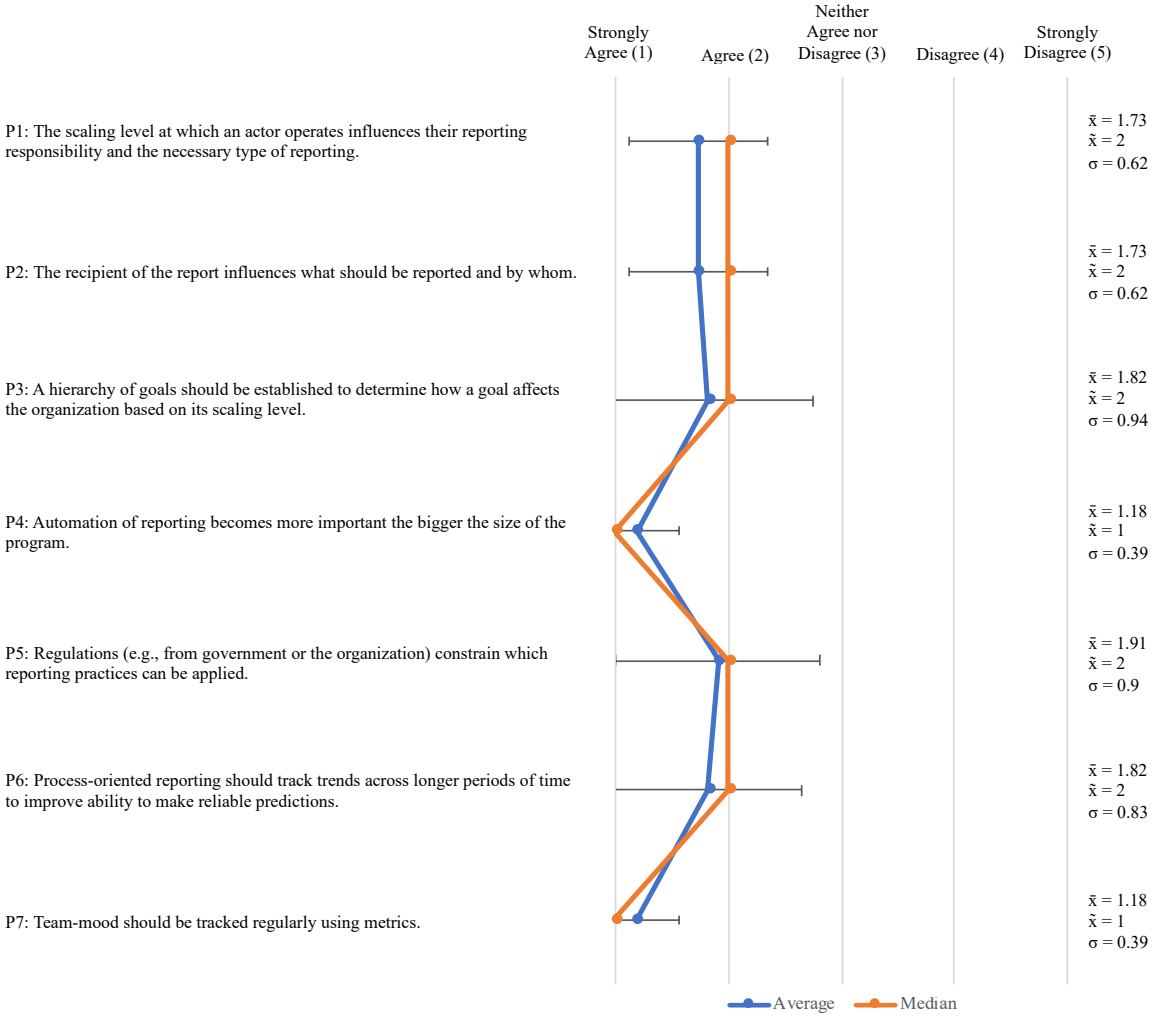


Figure 6.1.: Evaluation results of the general propositions

this aggregation named by participants include lower granularity of reported information, or usage of different KPIs on different scaling levels. This feedback corroborates our findings in Section 4.4 of different metrics on different scaling levels. The second dimension identified by participants is the scope of reporting responsibility (*BE1, LM1, PO3*). Interviewees explained, that while *what* has to be reported by a specific actor remains similar on the different scaling levels, the scope of included programs, teams, or employees does change. This finding indicates that the static assignment of specific actors to specific types of reporting (e.g. Agile Masters to process-oriented reporting) as reported in literature (cf. Stettina and Schoemaker [59]) and observed by us in Chapter 4 is feasible, while the scope of a specific type of reporting may be different on different scaling levels.

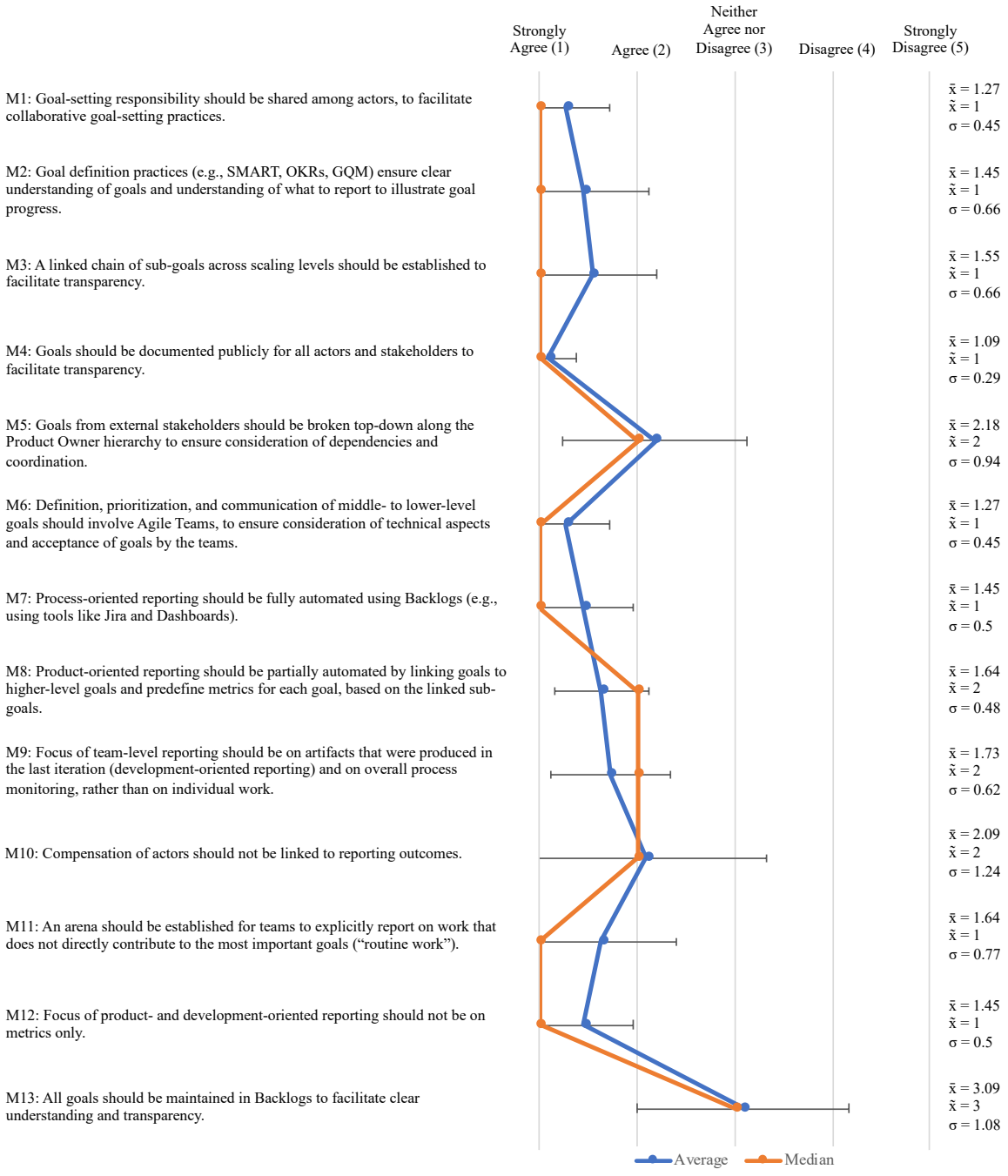


Figure 6.2.: Evaluation results of the mitigation propositions

P2: The recipient of the report influences what should be reported and by whom.

The evaluation of proposition P2 resulted in an average rating of 1.73 with a standard deviation of 0.62. Interviewees pointed out that reporter and recipient should mutually agree on what should be reported (*PO4, BE1, DEV1*), instead of relying on predefined reporting templates or framework suggestions. Participants stated that every report should have a clearly and transparently stated reason, and that recipients tend to prefer reports tailored to their task. However, interviewee *PO6* noted that the requestor of a report might be different from the actual recipient (e.g., proactive information report of Program Owners to Domain Owner). Thus this proposition might not be true for certain reports.

P3: A hierarchy of goals should be established to determine how a goal affects the organization based on its scaling level.

The evaluation of proposition P3 resulted in an average rating of 1.82 with a standard deviation of 0.94. Several participants pointed out that a hierarchy of goals should be established, and that sub-goals should be linked to higher-level goals that they are contributing to (*AM7, LM1, DEV1, PO4, PO5, PO6*). This is an interesting observation, as proposition P3 does not explicitly mention a linkage of goals, but merely a reflection of the organizational hierarchy in the goals. A linkage is explicitly suggested in proposition M3. Based on this feedback a consolidation of propositions P3 and M3 might be feasible. This interpretation is corroborated by interviewees *PO1* and *PO4*, who explicated that a hierarchy of goals on its own seems not useful unless combined with a linkage between sub-goals and higher goals. Besides this topic, interviewee *AM2* also suggested to limit the number of possible goals on a certain scaling level, similar to the limit of Objectives in the OKR practice.

P4: Automation of reporting becomes more important the bigger the size of the program.

The evaluation of proposition P4 resulted in an average rating of 1.18 with a standard deviation of 0.39. We observed that several interviewees were in favor of automation of reporting in general (*LM1, BE1, PO4*). Increasing size of a program is seen as an additional driver that further intensifies the need for automation, due to efforts quickly becoming infeasible otherwise (*DEV1, PO4, PO5, PO6*). Further, interviewee *AM7* mentioned that one should distinguish between *what was achieved* and *what was done*. Reporting of what was achieved can be realized using KPIs, according to *AM7*, and should be as automated as possible. Showing what was done cannot be achieved via automated reports, on the other hand. This distinction made by *AM7* is similar to the separation of process- and development-oriented reporting. Interviewee *AM2* mentioned that transparency also becomes more important with increasing size of the agile organization.

P5: Regulations (e.g., from government or the organization) constrain which reporting practices can be applied.

The evaluation of proposition P5 resulted in an average rating of 1.91 with a standard deviation of 0.9. Several interviewees mentioned that existing regulations are currently constraining the reporting practices that can be used in their programs (*AM2, AM7, LM1, BE1, PO3*). Most prominently, participants named the restriction of not being allowed to report on individual employees and contractors, which is seen to hinder transparency. Interviewee *LM1* described participation in controversial discussions on whether a report on the performance of a Product can be seen as an individual performance evaluation of the Product Owner — which would be violating existing regulations at the organization.

P6: Process-oriented reporting should track trends across longer periods of time to improve ability to make reliable predictions.

The evaluation of proposition P6 resulted in an average rating of 1.82 with a standard deviation of 0.83. Interviewees *AM7* and *PO5* cautioned that such a tracking over longer periods of time requires a certain stability in the organization and team composition, to ensure comparability between different points in time. Given this stability, participants acknowledged that the tracking over longer periods of time allows for identification of extraordinary effects and evolution of teams, which cannot be achieved with individual data points in time (*PO4, LM1, DEV1*). Interviewee *AM2* even emphasized continuous monitoring over tracking across longer periods of time, and *LM1* suggested to also apply this practice to other types of reporting.

However, interviewees pointed out that there might be metrics not feasible for tracking and comparison over time (*BE1, PO5*), and that individual data points in time might be valuable as well (*BE1, PO3*).

P7: Team-mood should be tracked regularly using metrics.

The evaluation of proposition P7 resulted in an average rating of 1.18 with a standard deviation of 0.39. Multiple interviewees emphasized that team mood influences team performance, and thus is important to consider (*LM1, BE1, DEV1, PO5*). Interviewees *AM6* and *AM7* mentioned team mood should be tracked using a metric in each Retrospective meeting. *PO5* urged to consider team mood in planning meetings as well. On the other hand, *AM6* emphasized that direct personal discussions with the teams should remain the primary means of understanding the current mood of the team.

M1: Goal-setting responsibility should be shared among actors, to facilitate collaborative goal-setting practices.

The evaluation of proposition M1 resulted in an average rating of 1.27 with a standard deviation of 0.45. Interviewees emphasized that good goal-setting is always a combination

of top-down and bottom-up interaction (*LM1, BE1, PO4*). No single actor's knowledge is sufficient to do all goal-setting independently (*BE1, LM1*), and teams should not only execute goals set on higher levels but contribute to the goal-setting with their technical expertise (*PO5*). *DEV1* and *PO1* mentioned that shared goal-setting increases team buy-in and transparency of goals. Participant *AM7* cautioned that shared goal-setting and aligned goals are a prerequisite in order for teams to be able to operate autonomously.

M2: Goal definition practices (e.g., SMART, OKRs, GQM) ensure clear understanding of goals and understanding of what to report to illustrate goal progress.

The evaluation of proposition M2 resulted in an average rating of 1.45 with a standard deviation of 0.66. In general, interviewees tended to be indiscriminate about which specific goal-setting practice to use, but rather emphasized that *one* goal-setting practice should be agreed on and standardized in a large-scale agile program (*LM1, BE1, PO4, PO5*). Agreeing on one specific goal-setting practice comes with several benefits, according to the participants. It reduces overhead as all actors and stakeholders know how goals should be defined (*LM1*), it ensures that goals are aligned both horizontally and vertically in the organization (*DEV1*), and it makes it possible to see trends in goal creation over time because goal definition is more consistent (*LM1*). *AM2* and *AM7* both stressed to use goal definition practices with a strong theoretical backing, and to refrain from too much customization of these practices.

M3: A linked chain of sub-goals across scaling levels should be established to facilitate transparency.

The evaluation of proposition M3 resulted in an average rating of 1.55 with a standard deviation of 0.66. *LM1* and *AM7* explained that the linkage of sub-goals to higher-level goals is a necessity to ensure a stringent implementation of an organization's overall strategy. It ensures that everyone is aware of why the defined goals are being pursued (*BE1, PO4*) and makes sure that goals actually contribute to a higher-level objective (*PO6*). However, multiple interviewees stressed that — while in general the linkage should be enforced — there may still be goals that cannot be directly linked to any higher-level objective (*AM2, LM1, PO3, PO5, PO6*). A commonly referred example are technical changes to the software product that are identified as necessary by the Agile Team but do not directly contribute to any higher goal, e.g., a version upgrade to an existing database (*AM2*). So, while a linkage of goals to higher-level goals facilitates a goal-oriented operation of the organization, it should not be forced on every necessary work-item in the daily work.

M4: Goals should be documented publicly for all actors and stakeholders to facilitate transparency.

The evaluation of proposition M4 resulted in an average rating of 1.09 with a standard deviation of 0.29. This is the best evaluation result of all propositions. Public documentation of goals, according to the participants, yields several benefits. It helps actors and stakeholders to identify who is working on similar objectives (*AM7, LM1*), and thus contributes to reducing redundant work (*AM7, LM1*). Further, it helps to align expectations of different stakeholders (*PO4*) and allows to identify a potential mismatch between goals and stakeholder expectations (*AM2*). Finally, it also facilitates transparency by clearly disclosing why actors and stakeholders are acting the way they do (*DEV1, PO1*).

M5: Goals from external stakeholders should be broken top-down along the Product Owner hierarchy to ensure consideration of dependencies and coordination.

The evaluation of proposition M5 resulted in an average rating of 2.18 with a standard deviation of 0.94. While interviewees *DEV1, PO3* and *PO4* agreed that a top-down distribution of external goals via the hierarchy of Product Owners does ensure that goals are properly defined and distributed, the evaluation shows that there is no clear consent on proposition M5. Proposition M5 was discussed in combination with proposition M6 in the interviews, as those two propositions are closely tied to each other. Most interviewees emphasized proposition M6 to be more important than proposition M5 for scaled agile development efforts. *AM2* suggested to determine whether goals should be broken down and distributed via the Product Owners depending on how specific a goal is. Very specific goals that can be clearly assigned to individual teams may not need such a top-down process, but can be directly discussed with the team.

This evaluation result suggests changes to proposition M5. The top-down process should only be applied in cases when stakeholders come up with broad, unspecific goals that cannot be clearly allocated to individual teams and thus entail dependencies and the need for coordination.

M6: Definition, prioritization, and communication of middle- to lower-level goals should involve Agile Teams, to ensure consideration of technical aspects and acceptance of goals by the teams.

The evaluation of proposition M6 resulted in an average rating of 1.27 with a standard deviation of 0.45. In contrast to the top-down part of the goal-setting process described in proposition M5, the bottom-up involvement of the Agile Teams in goal-setting proposed in M6 resulted in a more positive evaluation by practitioners. One important reason why teams have to be involved is the lack of technical expertise on higher organizational levels (*PO1, PO3*). Interviewees *BE1* and *PO5* emphasized that goals should never be committed to stakeholders without involvement of the Agile Teams. While Agile Teams should

be involved in goal-setting, the final say in prioritization of goals in large-scale agile development should remain with the Product Owner (*AM2, LM1*). Based on this, *LM1* also stressed that while teams should be part of the goal-setting process, they should not be held accountable for goal-setting.

M7: Process-oriented reporting should be fully automated using Backlogs (e.g., using tools like Jira and Dashboards).

The evaluation of proposition M7 resulted in an average rating of 1.45 with a standard deviation of 0.5. Interviewees *LM1, BE1, DEV1, PO3,* and *PO5* emphasized that automation should be pursued as much as possible to reduce manual effort of process-oriented report generation. *PO5* even thinks that any additional, manually created report besides the actual development Backlog is unnecessary overhead, because the Backlog should represent the current state of development at any given point in time. However, *AM2, PO1, PO4,* and *PO5* cautioned to always consider context information when interpreting and discussing automatically generated reports based on metrics. The team or actor in charge of a metric should always be involved in discussions of it, to be able to provide background and context information if needed. Additionally, automating the generation of a specific report is always a trade-off between the stability of said report and the effort of automating it (*LM1, DEV1*). Reports that remain consistent for longer periods of time (e.g., process-oriented reports) are more suitable to be automated than reports that are different each time (e.g., development-oriented reports).

M8: Product-oriented reporting should be partially automated by linking goals to higher-level goals and predefine metrics for each goal, based on the linked sub-goals.

The evaluation of proposition M8 resulted in an average rating of 1.64 with a standard deviation of 0.48. Participants *LM1, BE1,* and *PO6* explained that product-oriented reporting should be a mixture of quantitative and qualitative means of reporting, as well as manual and automatic generation. *DEV1* and *PO3* also explicitly agreed that progress of higher-level goals should be derived from linked sub-goals.

M9: Focus of team-level reporting should be on artifacts that were produced in the last iteration (development-oriented reporting) and on overall process monitoring, rather than on individual work.

The evaluation of proposition M9 resulted in an average rating of 1.73 with a standard deviation of 0.62. *PO1, PO3,* and *PO4* explained that only the team result is what counts and should be reported. Since the Agile Team is self-managed, it is also up to the team how work is coordinated internally. Further, *PO1* and *PO5* referenced existing work regulations that do not allow to report on individuals (cf. proposition P5). On the other hand, for

participants *AM6* and *PO5* it is still important to highlight exceptionally good individual contributions in the team-level reporting.

M10: Compensation of actors should not be linked to reporting outcomes.

The evaluation of proposition M10 resulted in an average rating of 2.09 with a standard deviation of 1.24. This proposition was evaluated rather controversially by the practitioners. While several interviewees agreed that compensation should not be linked to reporting outcomes (*AM2, AM6, AM7, LM1, DEV1, PO1, PO3, PO6*), some also disagreed with the statement (*BE1, PO4, PO5*). *AM2, LM1,* and *PO5* emphasized that compensation based on individual performance should be avoided to improve team performance, which is in line with goal-setting theory. Further, *AM7, LM1,* and *PO5* explained that linking compensation to KPIs tempts actors to game the system and manipulate those KPIs, leading to local optimization instead of desired global optimization (*PO5*). However, using a similar argumentation, higher-level team or organization goals can be used for compensation, according to *AM2* and *PO4*, because this would yield global optimization. Finally, *LM1* argues that an implicit linkage between reporting outcomes and monetary compensation can never be avoided, since well performing individuals tend to be paid more and be promoted more quickly within the organization.

Based on these evaluation results we derive the need to make proposition M10 more precise. The suggestion by practitioners that compensation can indeed be linked to higher-level (program, portfolio, organization) goals should be considered in M10.

M11: An arena should be established for teams to explicitly report on work that does not directly contribute to the most important goals ("routine work").

The evaluation of proposition M11 resulted in an average rating of 1.64 with a standard deviation of 0.77. Several interviewees think that there should be more positive recognition for employees doing their normal work very good (*AM2, LM1, PO3, PO5*). Nevertheless, multiple participants also emphasized that Agile Teams should focus their work efforts on the most important topics, and if teams are working on topics that are not part of any goal, goals seem to not represent the priorities of the organization correctly (*AM7, PO4, PO6*). *AM7* emphasized to not use such an arena as an excuse why work has been focused on less important topics. *DEV1* suggested to allow teams to decide whether such an arena is established or not.

M12: Focus of product- and development-oriented reporting should not be on metrics only.

The evaluation of proposition M12 resulted in an average rating of 1.45 with a standard deviation of 0.5. Almost all participants mentioned that product- and development-oriented reporting should not focus on metrics only, but has to also comprise context information

and direct human interaction (*AM2, AM6, LM1, BE1, DEV1, PO3, PO4, PO5*). Only *AM7* raised a concern, that focus on qualitative reporting may leave too much room for justification and excuses why certain metrics are bad. Hence, a balance has to be achieved between rigorous evaluation using metrics and enough qualitative information to be able to interpret the metrics correctly.

M13: All goals should be maintained in Backlogs to facilitate clear understanding and transparency.

The evaluation of proposition M13 resulted in an average rating of 3.09 with a standard deviation of 1.08. This proposition was evaluated very controversially and received the worst evaluation result of all propositions. During evaluation interviews, we observed two basic points of view regarding this statement. On the one hand, several observed cases see the Backlog as a central point in their development effort where all objectives (i.e., goals as well as requirements) should be stored (*DEV1, PO6*). On the other hand, multiple participants are of the opinion that goals and requirements should be strictly separated, and hence that goals should not be documented in a Backlog (*AM2, PO4, PO5*). Further, interviewees *LM1, BE1,* and *PO5* acknowledged that it also depends on the definition of a Backlog.

Ultimately participants from both sides, however, agreed that there has to be at least a connection between the development Backlogs and the organizational goals (*AM2, AM6, LM1, BE1, PO1, PO4*), and that goals should be documented digitally (*AM2, AM7, PO1, PO4*). They emphasized that Backlog Items — especially requirements that are maintained in Backlogs — have to be linked to the respective organizational goal that they are contributing to. Hence, this can be interpreted as an extension of proposition M3 to also cover development requirements.

6.3. Adjustments based on Evaluation

Propositions

Based on the evaluation results summarized in the previous section, we derived several potential areas of improvement to the propositions.

We observed that for practitioners the hierarchy of goals proposed in P3 is not of high value on its own. Rather, practitioners only see value in such a hierarchy if combined with linking goals to higher-level goals they contribute. This is what is proposed in M3. Considering this evaluation observation, we suggest to merge propositions P3 into proposition M3. This yields the following new proposition **M3**: *A hierarchy of goals should be established. Goals should be linked to the goals on the next higher level that they are contributing to, to facilitate transparency (liked chain of sub-goals).*

Next, interviewees did not evaluate the top-down process of goal-setting described in M5 very positively. We observed, that such a top-down process is only valuable to agile

practitioners in cases where stakeholders come up with broad, unspecific goals that cannot clearly be allocated to individual teams and thus entail dependencies and the need for coordination. Based on this feedback we make the following adjustment to proposition **M5**: *Goals from external stakeholders that cannot be clearly assigned to individual teams should be broken top-down along the Product Owner hierarchy to ensure consideration of dependencies and coordination.*

In the evaluation interviews we also learned that the topic of compensation is discussed controversially in large-scale agile environments. The general proposition M10 — compensation in general should not be linked to reporting outcomes — was agreed to by part of the interviewees while rejected by another part. Interviewees instead suggested to link compensation to higher-level goals. While lower-level or individual goals are not suitable to link compensation to, because this would lead to local optimization, higher-level goals are deemed suitable by participants. Using this suggestion, we rephrase proposition **M10**: *Compensation of actors should not be linked to reporting outcomes of lower-level or individual goals. Instead, use only higher-level goals for variable compensation.*

Finally, proposition M13 was also discussed controversially and received the worst evaluation result. We observed that there was no common agreement as to whether goals should or should not be maintained in Backlogs. Hence, we delete proposition M13 from the list of propositions. Additionally to this feedback, we also observed that practitioners do find consensus in linking Backlog Items to the goals that they are contributing to. Proposition M3 can therefore be adjusted to also cover Backlog Items in the linked chain of sub-goals. We make the following changes to **M3**: *A hierarchy of goals should be established. Goals and Backlog Items should be linked to the goals on the next higher level that they are contributing to, to facilitate transparency (liked chain of sub-goals).*

General Aspects

At the end of the interviews, we asked the participants whether they had any further topic or comment related to the interview that might be of interest to the research project. With this question we sought to uncover potential topics that were not yet covered.

One topic that was mentioned by several interviewees (*PO1, PO3, PO4, PO5*) is the need for regular inspect and adapt cycles for goals, as well as for the goal-setting and reporting processes. The interviewees stressed that in an large-scale agile environment it is important to not only be agile in product development, but also to be agile in the design of work processes and goals. Thus, they see the need to regularly discuss the goal-setting and reporting processes in retrospectives, in order to identify potential problems or opportunities for improvement. This aspect is partially reflected in the *analyze and intervene* step of our process model. In this step — as described in Section 5.3 — we recommend several interventions on **goals** based on the analysis results. Using the feedback we collected in the evaluation, this list of possible interventions should be extended by an intervention focused on **processes**. We suggest to add an intervention *adjust* also for the goal-setting processes, similar to the *adjust* intervention for the reporting process. Adjustments to the

goal-setting process might be necessary if goals cannot be reported towards properly, if goals lack quality in definition, if goals are repeatedly missed, and other cases.

Another topic that was mentioned by interviewees *PO4* and *PO5* is limiting the number of goals at a certain organizational level. At one level (e.g., for one team, or for one Product) the maximal number of goals should be limited per iteration, according to interviewees. This is similar to how e.g., the OKR approach limits the number of Objectives and the number of Key Results per Objective. However, while we see that limiting the number of goals may increase focus on the most important topics, we suggest to not do so in general. We rather think this is a decision that should be taken based on the specific goal definition practice (e.g., OKRs) used by a program.

7. Discussion

This section summarizes the key findings that we learned from the presented research project. It also reviews quality considerations and potential limitations of the master's thesis.

7.1. Key Findings

Reporting responsibility is separated and dependent on the scaling level in large-scale agile environments

We find that reporting responsibility in agile development is split into three parts, development, product, and process reporting, which is in line with existing literature (cf. [59]). We further find that in *large-scale* agile programs different types of reporting are of changing importance depending on the scaling level. While process- and product-oriented reporting are found to be of equal importance across scaling levels, development-oriented reporting is mostly relevant on lower organizational scaling levels. In contrast to literature (cf. [59]) we find that Product Owners are also sometimes involved in development-oriented reporting on lower organizational scaling levels.

The presence of process- and product-oriented reporting on all scaling levels at the case organization may correlate with our observation that the majority of goals can be categorized as *process* or *product* goals. It seems intuitive to us that pursuing many *process* and *product* goals poses a demand for process- and product-oriented reporting.

Bottom-up perspective becomes more important for goal-setting in large-scale agile organizations

In literature the bottom-up perspective of agile teams is often described as focusing on *how* to achieve or implement organizational goals, while the *what* of the goals is the responsibility of the top-down perspective of Product Owners and management (e.g., see [31, 53]). Our case study, and especially the evaluation of propositions M5 and M6, shows that in large-scale agile software development this distinction may not be suitable anymore. We observe that practitioners see the need for including the agile teams in all aspects of goal-setting up to the program-level in the organizations. This is due to increased complexity and scope of developed software products, which cannot be overseen by one of the two perspectives alone. Our findings indicate that integrated goal-setting where goals are

defined, selected, and prioritized collaboratively seems suitable to cope with these challenges.

Most goals are set at the portfolio- and program-level at the large-scale agile case organization

We observed that at the case organization most goals were set at the portfolio- and program-level. On portfolio-level we documented 21 goals, and on program-level we documented 26 goals. We identified one goal that was only set at the team-level, and seven goals that were relevant enterprise-wide. This may indicate that in large-scale agile organizations the middle levels (portfolio and program) are of higher relevance for goal-setting than the lowest (team) and highest (enterprise) level.

We further find that both *process* and *product* goals are roughly equally often pursued on portfolio and on program level. This observation stands in contrast to the findings by Korpivaara et al. [31], who find in their study that process efficiency objectives are more often set at the program-level compared to higher levels in organizations.

Metrics are of common use on all scaling levels at the large-scale agile case organization

In the case study we documented 33 metrics used by the observed programs. Metrics at the case organization are used on all scaling levels, with most of them being applied at the program-level. Several individual metrics are also tracked on multiple scaling levels. However, we did not observe any metric that is used by all participating programs of the study. The choice of used metrics is highly individual to the programs and their current situations, and adjusted by them over time. This may indicate that there is no set of metrics that should always or never be used. Instead, metrics are used to satisfy concrete information needs that are dependent on the current situation and goals.

Many challenges are not specific to large-scale agile environments

A considerable part of the identified challenges of goal-setting and reporting are not specific to large-scale agile software development. Instead, those challenges are general challenges that arise with goal-setting and reporting in (agile) organizations, such as dependencies, regulations, reporting on qualitative goals, and others. As a result, also parts of the propositions that we formulated, and parts of the process models we documented, are not specific to only large-scale agile software development. Instead, they might also be applicable to other software development environments. This shows that while agile methodologies change parts of the working processes in large-scale organizations quite drastically compared to traditional methodologies, not all aspects and challenges are affected. Also, agile methodologies certainly are no silver bullet for solving all challenges

in large organizations. This finding is in line with extant literature on challenges of large-scale agile software development and transformations (e.g., cf. [14, 62]).

In practice, the distinction between goals and requirements is blurred

Goals are often seen as abstract concepts that are out of reach of "ordinary" employees in large software development programs. Every objective "ordinary" employees deal with is often simply considered a requirement. Higher-level Backlog Items (e.g., Epic, Saga) are considered to be requirements by some, while others consider them to be goals. User Stories are mostly considered to be requirements that contribute to or implement higher-level goals. This is clearly reflected in the evaluation results of proposition M13, where we observed two opinions on whether goals can be Backlog Items or not. Based on this observation organizations should try to transparently document their goals, and clearly show for requirements if and to which goals they contribute. We captured this suggestion in several of our propositions (e.g., M3, M4).

Development-oriented reporting is not perceived as reporting in agile development

In the evaluation interviews, when explaining the split of reporting into three types of responsibility (development, product, and process reporting) as described in literature, several interviewees expressed skepticism. They provided us the feedback that, in their opinion, development-oriented reporting such as Review meetings, System Demos, etc., are not something they would typically consider under the term reporting. We learn that the term is coined quite negatively in the case organization. Many interview participants associated reporting directly with management control. This might be the case because most of the observed case programs transitioned from traditional software development to agile methodologies, rather than being set up with agile methodologies from scratch. In the traditional development processes reporting was indeed linked to management control at the case organization.

Story Points are not used for higher-level reporting in large-scale agile development

At the studied large-scale agile case organization we did not observe systematic usage of Story Points for reporting processes above the team-level. Story Points are used to estimate the effort of specific work items in the development Backlog. We would have expected to find usage of Story Points also for reporting purposes. However, at the case organization Story Points are mostly used for team internal discussions of work packages and effort estimation. Team-level reporting was also observed to use Story Points. On higher-levels, however, we did not observe systematic use of Story Points for reporting.

Control was not identified as a reason behind reporting

Controlling of work was not identified as an explicit reason behind any type of goal-setting or reporting at the case organization. We would have expected control to be among the reasons for certain reporting approaches, especially because control theory is a well studied area of research (e.g., cf. Dreesen et al. [18]). However, management control was still perceived by interviewees as a challenge for team autonomy. This indicates that communication and culture are essential in agile environments, to clearly define what is the actual purpose behind goals and reports.

7.2. Limitations

This section reviews the quality of the presented research and discusses potential threats to validity and reliability. Yin [68] formulates four principles for data collection in case study research. We applied these principles to the case study that we conducted as part of this thesis. The first principle recommends to use multiple sources of evidence [68]. We realized this principle by using first degree data and third degree data for our case study. First degree data was collected via the interviews we conducted with practitioners at the case organization. Third degree data was collected via documentation, internal Wiki pages, presentations, and other sources collected at the case organization. The second principle recommends to maintain a case study database [68]. We maintained such a database, which contains all the collected data from the case organization, relevant literature, transcripts, data codings, and other researcher generated artifacts. The third principle recommends to maintain a chain of evidence [68]. We ensured a stringent chain of evidence by coding and analyzing the collected data using a defined structure of codes, following the methodology of Miles et al. [41]. Finally, the fourth principle recommends to be careful when using data from electronic sources [68]. We only collected electronic data from internal sources at the case organization to ensure they are authentic and reliable.

Following the definitions by Runeson and Höst [52] we differentiate between construct validity, internal validity, external validity, and reliability. **Construct validity** considers whether the studied phenomena really represent what should be investigated based on the research questions [52]. To address potential threats to construct validity, as suggested by Yin [68], we used multiple sources of data (data triangulation) and established a chain of evidence. Additionally, we interviewed participants with diverse roles and experience in software development to ensure to gather insights from multiple perspectives. **Internal validity** considers whether the findings of the research actually represent the studied phenomena authentically [52]. To address potential threats to internal validity, we conducted a preparation meeting with each case study participant before the actual interview. This ensures a common understanding of concepts and terms between researchers and participants. **External validity** considers whether the findings of the research can be generalized to situations beyond the specific case at hand [52]. We conducted a single case, embedded

case study. While only one organization was studied, multiple programs at the organization were considered, each of which create their working processes independently from each other. However, our findings are strictly limited to the scope of large-scale agile software development. **Reliability** considers whether the same study could be replicated by another researcher later on [52]. To address potential threats to reliability, we described our coding system in detail and listed all interview questionnaires in the appendix of this thesis. However, while we did cross-case analysis between the studied programs, the findings are specific to the case organization. Another organization may be using very different approaches for goal-setting and reporting. Thus the same study may be replicated by another researcher using our documentation, but results may differ quite significantly.

8. Conclusion

The last chapter of this thesis summarizes our findings to answer the research questions, and gives an outlook to possible future research to be conducted in the area of goal-setting and reporting in large-scale agile software development.

8.1. Summary

In this section we summarize the answers to the research questions of the thesis.

RQ1. How are goals in large-scale agile software development established and reported at the case organization?

At the studied case organization, a large German car manufacturer, we identified 51 goals and structured them into the five categories *Product*, *Process*, *Resource*, *Strategic*, and *Legal, Security & Compliance*. We documented 20 goal-setting practices across the seven programs we studied in the initial case study. On the reporting side, we documented the application of 33 metrics in total, with 12 metrics being applied on team-level and 26 metrics being applied on program- and portfolio-level. 26 reporting practices have been documented from the studied programs.

In summary, the goal-setting and reporting at the case organization are dominated by an obligatory, organization-wide, standardized process. This process has been established before agile methodologies were used at the organization, and has been adapted to them over time. Apart from this process, goal-setting and reporting varies between the studied programs, and no further standardized practices are applied. Especially between the different programs we observed different approaches to goal-setting and reporting, since programs choose those internal practices autonomously. Programs are using custom documentation and tools for maintaining their goals, as well as different ways to assemble and document their reports. The most common goals we observed are system stability and legal compliance, while most other goals are specific to individual programs. The majority of the goals we documented can be categorized as *process* or *product* goals, and most goals are set at portfolio- and program-level. Most commonly programs are using wiki pages, Backlogs, and linkage to higher-level goals for goal-setting. For reporting, traffic light colors, sprint reviews, and all kinds of Dashboards (e.g., in Jira) are applied most often.

RQ2. What are challenges and reasons for establishing and reporting goals in large-scale agile software development?

To answer the second research question, we documented 19 challenges from the case organization. 11 challenges refer to goal-setting, while 9 challenges refer to reporting — with one challenge referring to both categories. The most commonly named challenges include perceived management control limiting team autonomy, external dependencies that limit team autonomy, and prioritization conflicts between goals.

Also, we identified 14 reasons behind the chosen reporting practices, and 13 reasons behind the goals that we documented for the first research question. We structured the reasons for the reporting practices into the four categories *Information Needs* (4), *Agile Values and Principles* (4), *Regulations* (2), and *Size of Program / Organization* (2). The most common reasons named for reporting include selecting reports based on the stakeholders and based on what is compliant with organizational regulations and governance. The reasons behind the goals can be structured into *Internal Needs* (6), *Customer Needs* (3), *Strategy of Organization* (2), and *Regulations* (1). Most commonly, the reasons behind why programs pursue goals include requests directly made by a customer and having to fulfill governmental regulations.

RQ3. How can these challenges for establishing and reporting goals in large-scale agile software development be addressed?

To address the identified challenges, we formulated 20 propositions. Seven of these propositions are general statements on goal-setting and reporting, while the remaining 13 propositions are intended to address (most of) the identified challenges. These propositions formalize our learnings from literature and the case study. They describe a theoretical basis for the process models for goal-setting and reporting that we developed. The process models in turn describe which of our propositions are relevant in which steps of goal-setting and reporting, and how they all fit together. The evaluation of the 20 propositions shows that in general practitioners agree with our theory on goal-setting and reporting. The statement that received the best evaluation — with all but one interviewees strongly agreeing — proposes to document all goals publicly for all actors and stakeholders. This shows that transparency is key for agile practitioners in large-scale organizations. The proposition to document all goals in Backlogs, on the other hand, received the worst evaluation result. We learned that practitioners do not find a consensus as to how goals could be maintained in Backlogs and often prefer to keep requirements and goals clearly separate.

After the evaluation, based on the collected feedback we made adjustments to two propositions, merged one proposition into another one, and deleted one proposition. The final set of propositions — in particular the remaining 12 propositions that address identified challenges — form the result for the third research question.

8.2. Outlook

In this project we implemented one cycle of the Action Design Research approach. We pointed out potential areas of improvement that we identified in the evaluation in Chapter 6. Thus, future work should use the evaluation results to conduct a second cycle of the Action Design Research approach, to further refine the propositions and process models. Further, the designed processes should be actually implemented at an active organization as part of the intervention of a future Action Design Research cycle. Such an implementation was out of scope of this master's thesis due to time constraints. Apart from additional Action Design Research cycles, the findings should be corroborated or refuted by further similar research projects with other case organizations. By investigating goal-setting and reporting in other large-scale agile organizations, our findings could be better generalized by future work.

A. Appendix

A.1. Case Study Interviews

First Part: General Information

1. What is your large-scale agile development (stakeholder) role?
2. How long have you been working in the field of software development?
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?
7. Does your company operate internationally?
8. Which sector does your company operate in?
9. How many employees does your company have?
10. How large is the development group / organization?
11. How many teams are co-located and how many teams are distributed?
12. In which countries are the teams located?
13. Which scaling agile frameworks are you using for the software development process?
14. To which level do you scale agile practices?
15. Are we allowed to contact you again for further research?

Second Part: Goals in Large-Scale Agile Development

1. What are the most important organization-specific goals that you are pursuing in your agile environment?
2. How do you categorize this goal?
3. What are the reasons that you are using this goal?
4. How important is this goal?
5. To which stakeholders is the goal relevant?
6. Which stakeholders were involved in defining the goal?
7. How were the individual teams involved in defining this goal?
8. Did you use specific techniques for defining this goal?
9. How is the goal documented and communicated?
10. Which challenges did you face when establishing this goal for / with the teams?

11. How do you approach these challenges?
12. Which actions can be (ideally) implemented to avoid these challenges?

Third Part: Reporting Team Progress on Goals

1. How do the teams report their progress towards the goal?
2. How do you categorize this reporting?
3. Why did you choose this type of reporting?
4. Which stakeholder(s) are responsible for this reporting?
5. When and how often is the progress reported?
6. Are you using (planning to use) any tools for the reporting?

Fourth Part: Reporting Overall Progress on Goals

1. How is overall program / product progress towards the goal reported?
2. How do you categorize this reporting?
3. Why did you choose this type of reporting?
4. How are individual team reports used for reporting overall program / product progress?
5. Which stakeholder(s) are responsible for reporting overall program / product progress?
6. When and how often is overall program / product progress reported?
7. Which stakeholder(s) are interested in the report of overall program / product progress?
8. What are the biggest challenges you are facing in reporting of overall program / product progress?
9. How do you approach these challenges?
10. Which actions can be (ideally) implemented to avoid these challenges?
11. Are you using (planning to use) any tools for reporting overall program / product progress?

Fifth Part: Discussion

1. How can research support the usage of metrics in the industry?
2. Do you want to add any further information, comment, or a topic that we missed?

A.2. Evaluation Interviews

First Part: General Information

1. What is your large-scale agile development (stakeholder) role?
2. How long have you been working in the field of software development?
3. How long have you been working in agile software development?
4. How long have you been working in large-scale agile software development?

Second Part: General Propositions

The approach considers processes for goal-setting and reporting. It comprises the actors Portfolio / Program / Product Owner, Agile Master, Line Manager, and Agile Team. These actors are working on the scaling levels Portfolio, Program, and Team. All actors are part of the overall organization. Customers, the Board of Directors, top-management, and other departments are considered external stakeholders to the large-scale agile development environment.

Questions under (a) use Likert scale answers: Strongly agree, Agree, Neither agree nor disagree, Disagree, Strongly disagree.

Proposition P1:
- a. The scaling level at which an actor operates influences their reporting responsibility and the necessary type of reporting.
- b. Why do you think the scaling level does (not) have an influence on the actors' reporting responsibility and necessary type of reporting?

Proposition P2:
- a. The recipient of the report influences what should be reported and by whom.
- b. Why do you think the recipient of the report does (not) have an influence on what should be reported and by whom?

Proposition P3:
- a. A hierarchy of goals should be established to determine how a goal affects the organization based on its scaling level.
- b. Why do you think a hierarchy of goals should (not) be established?

Proposition P4:
- a. Automation of reporting becomes more important the bigger the size of the program.
- b. Why do you think that automation of reporting does (not) become more important the bigger the program?

Proposition P5:
- a. Regulations (e.g., from government or the organization) constrain which reporting practices can be applied.
- b. Why do you think that regulations do (not) constrain which reporting practices can be applies?

Proposition P6:
- a. Process-oriented reporting should track trends across longer periods of time to improve ability to make reliable predictions.
- b. Why do you think process-oriented reporting should (not) track trends across longer periods of time?

Proposition P7:
- a. Team-mood should be tracked regularly using metrics.
- b. Why do you think team-mood should (not) be tracked regularly?

Third Part: Propositions to Address Challenges

Proposition M1:
a. Goal-setting responsibility should be shared among actors, to facilitate collaborative goal-setting practices.
b. Why do you think goal-setting responsibility should (not) be shared among actors?

Proposition M2:
a. Goal definition practices (e.g., SMART, OKRs, GQM) ensure clear understanding of goals and understanding of what to report to illustrate goal progress.
b. Why do you think goal definition practices do (not) facilitate clear understanding of goals and reporting?

Proposition M3:
a. A linked chain of sub-goals across scaling levels should be established to facilitate transparency.
b. Why do you think such a linked chain of sub-goals does (not) facilitate transparency?

Proposition M4:
a. Goals should be documented publicly for all actors and stakeholders to facilitate transparency.
b. Why do you think goals should (not) be documented publicly to facilitate transparency?

Proposition M5:
a. Goals from external stakeholders should be broken top-down along the Product Owner hierarchy to ensure consideration of dependencies and coordination.
b. Why do you think goals from external stakeholders should (not) be broken top-down along the Product Owner hierarchy?

Proposition M6:
a. Definition, prioritization, and communication of middle- to lower-level goals should involve Agile Teams, to ensure consideration of technical aspects and acceptance of goals by the teams.
b. Why do you think Agile Teams should (not) be involved in definition, prioritization, and communication of middle- to lower-level goals?

Proposition M7:
a. Process-oriented reporting should be fully automated using Backlogs (e.g., using tools like Jira and Dashboards).
b. Why do you think process-oriented reporting should (not) be fully automated?

Proposition M8:
a. Product-oriented reporting should be partially automated by linking goals to higher-level goals and predefine metrics for each goal, based on the linked sub-goals.
b. Why do you think product-oriented reporting should (not) be partially automated?

Proposition M9:
a. Focus of team-level reporting should be on artifacts that were produced in the last iteration (development-oriented reporting) and on overall process monitoring, rather

than on individual work.
   b. Why do you think focus of team-level reporting should not be on developed artifacts and process monitoring?

Proposition M10:
   a. Compensation of actors should not be linked to reporting outcomes.
   b. Why do you think compensation of actors should (not) be linked to reporting outcomes?

Proposition M11:
   a. An arena should be established for teams to explicitly report on work that does not directly contribute to the most important goals ("routine work").
   b. Why do you think such an arena should (not) be established?

Proposition M12:
   a. Focus of product- and development-oriented reporting should not be on metrics only.
   b. Why do you think focus of product- and development-oriented reporting should (not) be on metrics only?

Proposition M13:
   a. All goals should be maintained in Backlogs to facilitate clear understanding and transparency.
   b. Why do you think that all goals should (not) be maintained in Backlogs?

B. Appendix

B.1. Identified Goal-Setting Practices

Name	Description	Type	Source
SMART	Goals should be phrased specific, measurable, achievable, reasonable, and time-bound.	Definition	*STE1, BE1, AM5, PO1*
Objectives and Key Results	Goals should be phrased as qualitative Objectives. Each Portfolio and Program should have up to five Objectives per iteration. Each Objective, in turn, has up to four Key Results. Key Results are formulated quantitatively, such that the degree of their achievement can be measured. Once all Key Results are achieved, the Objective they belong to is achieved. The organization follows books by Doerr [17] and Wodtke [67].	Definition	*DEV1, LM1, STE1, AM2*
Product Press Releases	Goals are defined using fictional product press releases. The product press release describes the situation that is intended to be, once the goal has been reached. It is written together with the goal definition, and should help actors and stakeholders understand what the result of the goal will have to look like.	Definition	*AM5*
Weighted Shortest Job First (WSJF)	Goals that deliver the most value in the shortest period of time should be prioritized. It can be calculated by dividing the cost of delay by the effort size of the goal. WSJF is also described in the the SAFe framework [25].	Prioritization	*LM1, PO1*
Scoping	Yearly meeting to define milestones with target dates throughout the year. Milestones are based on the yearly goals of the portfolio, derived from the GMP goals. These milestones serve as input to the PI Plannings throughout the year. In the Scoping the Portfolio Owner, Program Owners, Product Owners, and all Agile Masters take part.	Meeting	*STE1*
Goal Management Process (GMP)	Process to break down goals from the Board of Directors along the hierarchy of Portfolio / Program / Product Owners. Goals are broken down from one level into sub-goals on the next level in a workshop between the respective Owners at the two scaling levels involved. Goals defined via this process have to be documented and reported using the GMP Sheet. The process focuses on defining *what* has to be done, but not *how*.	Process	*PO1, PO2, STE1, BE1, LM1, AM2, AM4*

Dual-Track Agile Goal-Setting	The dual-track approach is the definition of the next set of quarterly goals while the current quarter of the year is still running. The input to the process are the higher-level GMP goals on the one hand, as well as all additional goals gathered by Agile Teams, customers, partners, or other sources. It consists of alignment of goals by POs, documentation in the Backlog, research and ideation of potential solution directions, estimation, and the *Vision, Roadmap, and Direction (VRD) Focus Day*. In the VRD day, all actors and stakeholders select the goals for the next quarter collaboratively.	Process	*AM5*
PI Planning	The PI Planning serves the purpose of defining concrete goals for the PI, based on the milestones that were defined in the Scoping. The PI Planning consist of several steps. Two weeks before the PI Planning, the solution level defines and communicates the top milestones and priorities for the next PI ("solution vision"). Based on this solution vision, the Product Owners and their teams derive a product vision and according PI Objectives. All people of the Solution participate in the actual PI Planning, which lasts for one day. Every Product presents their planned Product Vision and PI Objectives. Each Product has a 1:1 "dependency session" with every other Product, to identify and discuss dependencies. After the PI, before each new Sprint the STE and POs have a shared session to discuss dependencies. The PI Planning is part of SAFe [25].	Meeting	*STE1, AM2, PO2, PO3*
APO / Saga Refinement	All goals are directed to the Area Product Owners / Program Owners. They conduct a recurring refinement session with all Program Owners. In this session they refine and break down the goals to the next lower level.	Meeting	*AM1, AM4, PO3*
Team Refinement	The goals received from the APO / Saga Refinement are refined and (if necessary) further broken down by the Agile Team. The Team Refinement is described in the Scrum Guide as Product Backlog refinement [56].	Meeting	*AM1, AM4*
Linked Chain of (Sub-)Goals	Each goal is linked to the goal it contributes to on the next higher organizational scaling level. This is done in different ways, e.g., using Backlogs, Confluence pages, PowerPoint slides.	Definition, Documentation	*STE1, BE1, LM1, AM1, AM2, AM4, AM5, PO2, PO3*
Cluster Backlog Items	Similar to Linked chain of (sub-)goals. Instead of linking, Backlog Items are clustered / grouped in the Backlog according to the goal they contribute to. Can be done just by grouping visually, or by applying labels to items.	Definition, Documentation	*AM2*
Individual Backlog for Business and Organizational Goals	In Program *C2*, goals relevant to business and goals relevant to organizational development are kept in separate Backlogs.	Documentation	*AM4*

Backlog	Not only requirements, but also goals of all types are stored as Backlog Items in various Backlogs. Some programs maintain one central Backlog for all items, while others maintain multiple Backlogs with different focus.	Documentation	*PO1, PO3, DEV1, STE1, AM1, AM4, AM5*
GMP Sheet	The GMP Sheet documents all goals at a certain organizational scaling level. For each goal a summary, current state, target state, deadline, and optional remarks are documented. This sheet is used both for documentation and reporting of goals.	Documentation	*PO1, BE1*
Central Wiki Page for Goals	Several programs are maintaining Wiki pages (e.g., in Confluence), where all current goals are documented. This typically is done with a simple list, often linking to the respective Backlog Items. Usually, the current state and responsible people are also documented in the list.	Documentation	*PO3, DEV1, LM1, STE1, AM2, AM3, AM5*
Weekly KPI Review	Program *B1* conducts a weekly KPI review meeting. All Program Owners and Product Owners participate. The meeting is intended to identify problems early and allow for discussion. Based on the identified problems and discussions, goals are set and / or refined.	Meeting, Communication	*PO1*
Product Jour-Fixe	After each PI Planning in Program *C1*, all Product Owners conduct a jour-fixe with their teams to discuss the committed priorities and goals. The meeting is intended for "post-processing" of the PI Planning and for clarifying what has to be done.	Meeting, Communication	*STE1*
PI Focus Topic	For each Program Increment, *PO3* defines a PI Focus Topic. This is the most important topic for the PI, usually a topic that has a (tight) hard deadline.	Prioritization	*PO3*
Domain Planning	The Domain Planning consist of several refinement sessions with the stakeholders. Based on the goals from the stakeholders and the Domain, Sagas are formulated in the Domain Backlog. These Sagas are then the basis for the individual PI Plannings later on. Stakeholders, Domain Owners, and Product Owners participate in the meeting.	Meeting	*PO3*

Table B.1.: List of identified goal-setting practices and meetings in the initial case study interviews

B.2. Identified Reporting Practices

Name	Description	Reporting Type	Level	Source
Informal ad-hoc reporting	Informal reporting based on ad-hoc meetings and communication in the office. This type of reporting does not have a standard format or medium.	All	All	*AM4*
Goal Management Process (GMP) Reporting	The standardized GMP comes with predefined templates (GMP Sheets) to be filled for reporting back to the top-management. The templates are mandatory to use at portfolio-level, but also often used on lower levels for consistency. A detailed description of the GMP and GMP Sheets is given in Sections 4.3.1 and 4.4.3.	Product	Portfolio (mandatory), all	*LM1, AM2, BE1, PO1*
Team Velocity Forecast	Each team has a Velocity Forecast that is calculated based on the average Velocity of the past five Sprints. It is used by teams and Programs to plan how much can be achieved in the next iteration cycle. The trend of the Velocity Forecast should be stable or going up; if it is decreasing investigation is necessary. The volatility of the Velocity Forecast can also be analyzed. If it is highly volatile, the team is not predictable, and one should investigate why.	Process	Program, Team	*B2, C2*
Saga per Customer	Program *D1* has an overarching Saga for each customer, to bundle all goals and requirements that refer to this customer in one place. This allows to easily see all work relevant for this customer in one place.	Process	All	*PO1*
Burndown Chart	The Burndown Chart shows the remaining amount of work over time during an iteration.	Process	All	*B2, B3*
Cumulative Flow Diagram	The Cumulative Flow Diagram gives an overview of the amount of Backlog Items in a certain state at a particular point in time.	Process	All	*B2, B3*
Dashboards	Dashboards (e.g., using Jira) are used to automatically track metrics and plot charts such as **Burndown Charts**, **Cumulative Flow Diagram**. They are used especially for process-oriented reporting by Agile Masters. For metrics that can be monitored using Dashboards, see Tables 4.5 and 4.6.	Process	All	All programs
Traffic Light	Status or progress towards a goal is reported by assigning a color. Green is assigned if everything is ok. Orange signals problems that can be solved by the reporting actor on itself. Red signals the need for escalation and external help. Red may also signal dependencies that are out of the influence of the reporting actor.	Product	All	*PO1, PO2, PO3, LM1, BE1, AM1, AM2*

Weekly solution jour-fixe	Weekly report of status of all Products part of the SAFe Solution. On the one hand, an automatic status report is generated from the status of all Backlog Items for each Product. On the other hand, each Product Owner assigns a Traffic Light evaluation to each of their goals. The two reports are compared, and potential discrepancies between them or problems are discussed. Evaluation by Product Owners is typically weighted higher than the auto-generated report.	Product	Portfolio, Program	*STE1, AM3*
Team Sprint Review	The standard Sprint Review meeting is used for development-oriented reporting by the Agile Teams. The developed artifacts are presented by the team and reviewed by the customer. The Sprint Review meeting is detailed in the Scrum Guide [56].	Development	Team	*AM1, AM2, AM3, AM5, BE1, LM1, PO3*
Overall Sprint Review	At the end of each Program cycle, all Agile Teams of the Program hold an overall Sprint review together. The Program Owner and — if available — the customers take part in the review. All goals, Backlog Items, and the developed artifacts themselves are reviewed. Also called Product Increment Review in some programs.	Development	Program, Team	*BE1, LM1, AM1, AM5, PO2, PO3*
PI Sync	The PI Sync takes place every three weeks. Each team has to evaluate the current state of their PI Objectives. State is evaluated as either Done, On Track, Partly, In Danger, Paused, Not Started, or N.A. Product Owners collect the states of all their teams in a central Wiki page.	Product	Program, Team	*STE1*
Milestone achievement report	In the beginning of a PI Planning, the milestones of the last PI are reviewed. Specific goals and Backlog Items are not considered, the Solution only evaluates whether the overall milestones have been achieved. If not, in the PI Planning they analyze the impact on the next PI and on the overall milestones that were defined in the Scoping.	Product	Portfolio, Program	*LM1, AM2*
Area Cycle Retrospectives	At each scaling level, after the end of the respective iteration cycle (Sprint, Program Cycle, Portfolio Cycle) there is a retrospective. The retrospective serves the purpose of continuous improvement, as described in the Scrum guide [56]. Issues from retrospectives can also be escalated to the next higher-level retrospective if they cannot be resolved on the current level. Thus, the retrospectives provide a line of escalation.	Product	All	*AM1*
Highlights and learnings of the quarter	Each quarterly report (in any form) at Program *C2* should contain a written statement that mentions the highlights and learnings of the past quarter.	Product	Program	*AM4*

OKR achievement report	In Program *C2* goals are defined and documented using OKRs. These OKRs are then also used to report back progress across the organizational levels, up to the portfolio-level. Each Objective is evaluated and reported based on its Key Results. Key Results, in turn, are evaluated using the linked Backlog Items.	Product	Portfolio, Program	*LM1*
Product goal review Wiki page	The **central Wiki page for goals** (see goal-setting practices) is also used for reporting the current achievement state of the goals. For each goal, a **Traffic Light** color is assigned to signal its state.	Product	Program	*B2, PO1, STE1*
Automated Backlog checks	Automatic checks of the Backlog (e.g., using Jira). These can be checks for whether acceptance criteria have been defined, parent / child goals have been linked, effort has been estimated, a description has been added, etc.	Process	All	*AM4, B1*
Team and Program Fitness Tracker	A central Wiki page where all Agile Teams of a Program self-report their "fitness". Teams themselves document their Predictability / Velocity Forecast, Velocity Forecast trend, and Velocity Forecast volatility. The page allows the Program Owner to keep an overview of all teams. The Predictability trend (Say-Do-Rate) is also tracked for the whole Portfolio over the last three cycles. The Portfolio Predictability is calculated as the median of all Programs' Predictabilities.	Process	Portfolio, Program, Team	*C2*
Mood survey	A recurring survey on the current mood of the employees in the Program. Can be easily implemented via macros, e.g., in Confluence. Results are reported to the Agile Masters and next higher organizational level, to correlate recent changes (e.g., in performance) with mood.	Process	Program, Team	*AM3*
Saga progress overview Excel	Products in program *B2* are using an Excel template to report progress on Sagas to the Domain Owner and higher-level management. For each Saga a Product is involved in the Excel shows how many of the linked sub-items of the Saga are already done or still open. The Excel sheet is filled by Product Owners after every Domain Cycle.	Product	Program	*PO2, PO3*
Weekly (Sub-)PO sync	Program *B2* has a weekly sync meeting, where Sub-Product Owners and Product Owners check in on each other. Sub-Product Owners report and discuss current issues or problems.	Product	Program, Team	*PO3*

Product Plan Review	Program B2 has a weekly meeting, where Sub-Product Owners and Product Owners participate. Sub-Product Owners report progress on all Epics that their teams are currently working on. If an Epic is postponed to another iteration, it receives a sticker on a whiteboard. If multiple stickers accumulate on an Epic it indicates need for action by Product Owners.	Product	Program	*PO2*
Operations Community	In a weekly meeting, Sub-Product Owners and Product Owners review the current state of operational KPIs. In case of system outages or unexpected KPI changes, the responsible PO has to explain what happened, whether and how it affected the customer, and what to do to avoid this incident in the future.	Process	Program	*PO2*
Risk Owner	Program *B2* has a dedicated role called Risk Owner. A Risk Owner is assigned a certain risk that was identified by the program, and has to monitor changes / events that may affect this risk or cause it to realize. If so, the Risk Owner has to report immediately to the Product Owners.	Product	Program	*PO2*
Agile Maturity Assessment	The Portfolio of which Program *C2* is a part of is using an Excel-based tool to evaluate the agile maturity of their programs. The assessment covers questions from the areas of roles, artifacts, events, tools, agile values, and company values. Each program has to answer these questions regularly to document their evolution regarding agile maturity.	Process	Program	*C2*

Table B.2.: List of identified reporting practices and meetings in the initial case study interviews

C. Appendix

C.1. Detailed Results of the Evaluation

Proposition	Strongly Agree (1)	Agree (2)	Neither Agree nor Disagree (3)	Disagree (4)	Strongly Disagree (5)
P1	4	6	1	0	0
P2	4	6	1	0	0
P3	5	4	1	1	0
P4	9	2	0	0	0
P5	4	5	1	1	0
P6	5	3	3	0	0
P7	9	2	0	0	0
M1	8	3	0	0	0
M2	7	3	1	0	0
M3	6	4	1	0	0
M4	10	1	0	0	0
M5	3	4	3	1	0
M6	8	3	0	0	0
M7	6	5	0	0	0
M8	4	7	0	0	0
M9	4	6	1	0	0
M10	5	3	0	3	0
M11	6	3	2	0	0
M12	6	5	0	0	0
M13	2	0	4	5	0

Table C.1.: Quantitative evaluation results per answer option for each proposition

Bibliography

[1] Ana I. Anton. "Goal Identification and Refinement in the Specification of Software-Based Information Systems". PhD Dissertation. Georgia Institute of Technology, 1997.

[2] Victor R. Basili, Gianluigi Caldiera, and H. Dieter Rombach. "The goal question metric approach". In: *Encyclopedia of software engineering* (1994), pp. 528–532.

[3] Victor R. Basili, Mikael Lindvall, Myrna Regardie, Carolyn Seaman, Jens Heidrich, Jürgen Münch, Dieter Rombach, and Adam Trendowicz. "Linking Software Development and Business Strategy Through Measurement". In: *Computer* 43.4 (2010), pp. 57–65. DOI: 10.1109/MC.2010.108.

[4] Thomas S. Bateman, Hugh O'Neill, and Amy Kenworthy-U'Ren. "A hierarchical taxonomy of top managers' goals". In: *Journal of Applied Psychology* 87.6 (2002), p. 1134. DOI: 10.1037/0021-9010.87.6.1134.

[5] Kent Beck. "Embracing Change with Extreme Programming". In: *IEEE Computer* 32.10 (1999), pp. 70–77. DOI: 10.1109/2.796139.

[6] Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, and Dave Thomas. *Manifesto for Agile Software Development*. 2001. URL: https://agilemanifesto.org (visited on 05/10/2021).

[7] Izak Benbasat, David K. Goldstein, and Melissa Mead. "The Case Research Strategy in Studies of Information Systems". In: *MIS Quarterly* 11.3 (1987), pp. 369–386. DOI: 10.2307/248684.

[8] Marthe Berntzen, Nils B. Moe, and Viktoria Stray. "The Product Owner in Large-Scale Agile: An Empirical Study Through the Lens of Relational Coordination Theory". In: *Agile Processes in Software Engineering and Extreme Programming*. Ed. by Philippe Kruchten, Steven Fraser, and François Coallier. Cham: Springer International Publishing, 2019, pp. 121–136. DOI: 10.1007/978-3-030-19034-7_8.

[9] Barry W. Boehm and Richard Turner. "Management challenges to implementing agile processes in traditional development organizations". In: *IEEE Software* 22.5 (2005), pp. 30–39. DOI: 10.1109/MS.2005.129.

[10] Martin P. Boerman, Zeeger Lubsen, Damian A. Tamburri, and Joost Visser. "Measuring and Monitoring Agile Development Status". In: *2015 IEEE/ACM 6th International Workshop on Emerging Trends in Software Metrics*. 2015, pp. 54–62. DOI: 10.1109/WETSoM.2015.15.

[11] The LeSS Company B.V. *LeSS Framework*. 2021. URL: https://less.works/less/framework/index.html (visited on 03/15/2021).

[12] Tjan-Hien Cheng, Slinger Jansen, and Marc Remmers. "Controlling and monitoring agile software development in three dutch product software companies". In: *2009 ICSE Workshop on Software Development Governance*. 2009, pp. 29–35. DOI: 10.1109/SDG.2009.5071334.

[13] Daniela S. Cruzes and Tore Dyba. "Recommended Steps for Thematic Synthesis in Software Engineering". In: *2011 International Symposium on Empirical Software Engineering and Measurement*. 2011, pp. 275–284. DOI: 10.1109/ESEM.2011.36.

[14] Kim Dikert, Maria Paasivaara, and Casper 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. DOI: 10.1016/j.jss.2016.06.013.

[15] Torgeir Dingsøyr, Tor E. Fægri, and Juha Itkonen. "What Is Large in Large-Scale? A Taxonomy of Scale for Agile Software Development". In: *International Conference on Product-Focused Software Process Improvement*. Cham, DE: Springer, 2014, pp. 273–276. DOI: 10.1007/978-3-319-13835-0_20.

[16] Torgeir Dingsøyr and Nils B. Moe. "Towards Principles of Large-Scale Agile Development. A Summary of the Workshop at XP2014 and a Revised Research Agenda". In: *Agile Methods. Large-Scale Development, Refactoring, Testing, and Estimation*. Cham, DE: Springer, 2014, pp. 1–8. DOI: 10.1007/978-3-319-14358-3_1.

[17] John Doerr. *Measure what matters: How Google, Bono, and the Gates Foundation rock the world with OKRs*. Penguin, 2018.

[18] Tim Dreesen, Phil Diegmann, and Christoph Rosenkranz. "The impact of modes, styles, and congruence of control on agile teams: Insights from a multiple case study". In: *Proceedings of the 53rd Hawaii International Conference on System Sciences*. 2020. DOI: 10.24251/hicss.2020.764.

[19] Tore Dybå, Dag I.K. Sjøberg, and Daniela S. Cruzes. "What Works for Whom, Where, When, and Why? On the Role of Context in Empirical Software Engineering". In: *Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*. ESEM '12. Lund, Sweden: Association for Computing Machinery, 2012, pp. 19–28. DOI: 10.1145/2372251.2372256.

[20] Norman E. Fenton. *Software Metrics: A Rigorous and Practical Approach*. Ed. by Shari L. Pfleeger. 2. ed. London: PWS Publishing, 1998.

[21] Quentin W. Fleming and Joel M. Koppelman. "Earned Value Project Management: A Powerful Tool for Software Projects". In: *The Journal of Defense Software Engineering* 19.6 (1998), pp. 19–23.

[22] Jody H. Gittell. "Relational coordination: Coordinating work through relationships of shared goals, shared knowledge and mutual respect". In: *Relational perspectives in organizational studies: A research companion* (2006), pp. 74–94.

[23] Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. "Design Science in Information Systems Research". In: *MIS Quarterly* 28.1 (2004), pp. 75–105. DOI: 10.2307/25148625.

[24] "IEEE Standard Adoption of ISO/IEC 15939:2007 - Systems and Software Engineering - Measurement Process". In: *IEEE Std 15939-2008* (2009), pp. 1–55. DOI: 10.1109/IEEESTD.2009.4775910.

[25] Scaled Agile Inc. *SAFe 5*. 2021. URL: https://www.scaledagileframework.com (visited on 03/13/2021).

[26] VersionOne Inc. *14th annual state of agile report*. 2020. URL: https://stateofagile.com/#ufh-i-615706098-14th-annual-state-of-agile-report/7027494 (visited on 03/13/2021).

[27] Project Management Institute. *Agile Practice Guide*. 1st edition. Boston, MA: Project Management Institute, 2017.

[28] Robert S. Kaplan and David P. Norton. *The Balanced Scorecard — Measures that Drive Performance*. 1992. URL: https://hbr.org/1992/01/the-balanced-scorecard-measures-that-drive-performance-2 (visited on 08/21/2021).

[29] Pertti Karhapää, Woubshet Behutiye, Pilar Rodríguez, Markku Oivo, Dolors Costal, Xavier Franch, Sanja Aaramaa, Michał Choraś, Jari Partanen, and Antonin Abherve. "Strategies to Manage Quality Requirements in Agile Software Development: A Multiple Case Study". In: *Empirical Software Engineering* 26.2 (2021), pp. 1–59. DOI: 10.1007/s10664-020-09903-x.

[30] Laurie J. Kirsch. "The Management of Complex Tasks in Organizations: Controlling the Systems Development Process". In: *Organization Science* 7.1 (1996), pp. 1–21. DOI: 10.1287/orsc.7.1.1.

[31] Ida Korpivaara, Tuure Tuunanen, and Ville Seppänen. "Performance Measurement in Scaled Agile Organizations". In: *Proceedings of the 54th Hawaii International Conference on System Sciences*. 2021, pp. 6912–6922. DOI: 10.24251/HICSS.2021.830.

[32] Eetu Kupiainen, Mika V. Mäntylä, and Juha 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. DOI: 10.1016/j.infsof.2015.02.005.

[33] Teemu Lappi, Teemu Karvonen, Lucy E. Lwakatare, Kirsi Aaltonen, and Pasi Kuvaja. "Toward an Improved Understanding of Agile Project Governance: A Systematic Literature Review". In: *Project Management Journal* 49.6 (2018), pp. 39–63. DOI: 10.1177/8756972818803482.

[34] Craig Larman. *Large-Scale Scrum*. Ed. by Bas Vodde. 1st edition. Boston, MA: Addison-Wesley Professional, 2016.

[35] Dean Leffingwell. *SAFe 4.5 Reference Guide: Scaled Agile Framework for Lean Enterprises*. 2nd edition. Boston, MA: Addison-Wesley Professional; Safari, 2018.

[36] Dean Leffingwell. *Scaling Software Agility: Best Practices for Large Enterprises*. Pearson Education, 2007.

[37] Rensis Likert. "A technique for the measurement of attitudes". In: *Archives of Psychology* (1932).

[38] Edwin A Locke and Gary P Latham. "Building a Practically Useful Theory of Goal Setting and Task Motivation: A 35-year Odyssey". In: *American psychologist* 57.9 (2002), p. 705. DOI: 10.1037/0003-066X.57.9.705.

[39] Edwin A. Locke and Gary P. Latham. "New Directions in Goal-Setting Theory". In: *Current Directions in Psychological Science* 15.5 (2006), pp. 265–268. DOI: 10.1111/j.1467-8721.2006.00449.x.

[40] John J. Marciniak. *Encyclopedia of Software Engineering*. 2nd. USA: Halsted Press, 2002.

[41] Matthew B. Miles, A. Michael Huberman, and Johnny Saldaña. *Qualitative Data Analysis. A Methods Sourcebook*. 3. ed. Los Angeles, USA: SAGE Publications, 2013.

[42] Nils B. Moe, Bjørn H. Dahl, Viktoria Stray, Lina S. Karlsen, and Stine Schjødt-Osmo. "Team Autonomy in Large-Scale Agile". In: *Proceedings of the Annual Hawaii International Conference on System Sciences (HICSS)*. AIS Electronic Library. 2019, pp. 6997–7006. DOI: 10.24251/HICSS.2019.839.

[43] Nils B. Moe, Viktoria Stray, and Rashina Hoda. "Trends and Updated Research Agenda for Autonomous Agile Teams: A Summary of the Second International Workshop at XP2019". In: *Agile Processes in Software Engineering and Extreme Programming – Workshops*. Ed. by Rashina Hoda. Cham: Springer International Publishing, 2019, pp. 13–19. DOI: 10.1007/978-3-030-30126-2_2.

[44] Ralf Müller, Miia Martinsuo, and Tomas Blomquist. "Project Portfolio Control and Portfolio Management Performance in Different Contexts". In: *Project Management Journal* 39.3 (2008), pp. 28–42. DOI: 10.1002/pmj.20053.

[45] Thomas Murphy and Kathryn Cormican. "Towards holistic goal centered performance management in software development: Lessons from a best practice analysis". In: *International Journal of Information Systems and Project Management* 3.4 (2015), pp. 23–36.

[46] Nilay Oza and Mikko Korkala. "Lessons Learned In Implementing Agile Software Development Metrics". In: *UK Academy for Information Systems Conference Proceedings 2012*. 38. 2012.

[47] Ken Peffers, Tuure Tuunanen, Marcus A. Rothenberger, and Samir Chatterjee. "A Design Science Research Methodology for Information Systems Research". In: *Journal of Management Information Systems* 24.3 (2007), pp. 45–77. DOI: 10.2753/MIS0742-1222240302.

[48] Drucker Peter. *People and Performance*. Routledge, 2012.

[49] Ken Power. "Definition of Ready: An Experience Report from Teams at Cisco". In: *Agile Processes in Software Engineering and Extreme Programming*. Ed. by Giovanni Cantone and Michele Marchesi. Cham: Springer International Publishing, 2014, pp. 312–319. DOI: 10.1007/978-3-319-06862-6_25.

[50] Abheeshta Putta, Maria Paasivaara, and Casper Lassenius. "Benefits and Challenges of Adopting the Scaled Agile Framework (SAFe): Preliminary Results from a Multivocal Literature Review". In: *Product-Focused Software Process Improvement*. Ed. by Marco Kuhrmann, Kurt Schneider, Dietmar Pfahl, Sousuke Amasaki, Marcus Ciolkowski, Regina Hebig, Paolo Tell, Jil Klünder, and Steffen Küpper. Cham: Springer International Publishing, 2018, pp. 334–351. DOI: 10.1007/978-3-030-03673-7_24.

[51] G. Regev and A. Wegmann. "Where do Goals Come from: the Underlying Principles of Goal-Oriented Requirement Engineering". In: *13th IEEE International Conference on Requirements Engineering (RE'05)*. 2005, pp. 353–362. DOI: 10.1109/RE.2005.80.

[52] Per Runeson and Martin Höst. "Guidelines for conducting and reporting case study research in software engineering". In: *Empirical Software Engineering* 14.2 (2009), pp. 131–164. DOI: 10.1007/s10664-008-9102-8.

[53] Ingo Schnabel and Markus Pizka. "Goal-Driven Software Development". In: *2006 30th Annual IEEE/NASA Software Engineering Workshop*. 2006, pp. 59–65. DOI: 10.1109/SEW.2006.21.

[54] Ken Schwaber. "SCRUM Development Process". In: *Proceedings of the 10th Annual ACM Conference on Object Oriented Programming Systems, Languages, and Applications*. 1995, pp. 117–134. DOI: 10.1007/978-1-4471-0947-1_11.

[55] Ken Schwaber. *The Nexus Guide*. 2018. URL: https://www.scrum.org/resources/nexus-guide (visited on 03/15/2021).

[56] Ken Schwaber and Jeff Sutherland. *The Scrum Guide*. 2020. URL: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf (visited on 05/10/2020).

[57] Maung K. Sein, Ola Henfridsson, Sandeep Purao, Matti Rossi, and Rikard Lindgren. "Action Design Research". In: *MIS Quarterly* 35.1 (2011), pp. 37–56. DOI: 10.2307/23043488.

[58] Dag I. K. Sjøberg, Tore Dybå, Bente C. D. Anda, and Jo E. Hannay. "Building Theories in Software Engineering". In: *Guide to Advanced Empirical Software Engineering*. Ed. by Forrest Shull, Janice Singer, and Dag I. K. Sjøberg. London: Springer London, 2008, pp. 312–336. DOI: 10.1007/978-1-84800-044-5_12.

[59] Christoph J. Stettina and Lennard 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.

[60] Viktoria Stray, Nils B. Moe, and Rashina Hoda. "Autonomous Agile Teams: Challenges and Future Directions for Research". In: *Proceedings of the 19th International Conference on Agile Software Development: Companion*. XP '18. Porto, Portugal: Association for Computing Machinery, 2018. DOI: 10.1145/3234152.3234182.

[61] Ömer Uludağ, Nina-Mareike Harders, and Florian Matthes. "Documenting Recurring Concerns and Patterns in Large-Scale Agile Development". In: *Proceedings of the 24th European Conference on Pattern Languages of Programs*. EuroPLop '19. Irsee, Germany: Association for Computing Machinery, 2019. DOI: 10.1145/3361149.3361176.

[62] Ömer Uludağ, Martin Kleehaus, Christoph Caprano, and Florian Matthes. "Identifying and Structuring Challenges in Large-Scale Agile Development Based on a Structured Literature Review". In: *2018 IEEE 22nd International Enterprise Distributed Object Computing Conference (EDOC)*. 2018, pp. 191–197. DOI: 10.1109/EDOC.2018.00032.

[63] Ömer Uludağ, Pascal Philipp, Abheeshta Putta, Maria Paasivaara, Casper Lassenius, and Florian Matthes. "Revealing the State-of-the-Art in Large-Scale Agile Development: A Systematic Mapping Study". In: *arXiv preprint arXiv:2007.05578* (2020).

[64] Axel van Lamsweerde. "Goal-Oriented Requirements Engineering: A Guided Tour". In: *Proceedings Fifth IEEE International Symposium on Requirements Engineering*. 2001, pp. 249–262. DOI: 10.1109/ISRE.2001.948567.

[65] Leo R. Vijayasarathy and Charles W. Butler. "Choice of Software Development Methodologies: Do Organizational, Project, and Team Characteristics Matter?" In: *IEEE Software* 33.5 (2016), pp. 86–94. DOI: 10.1109/MS.2015.26.

[66] Laurie Williams and Alistair Cockburn. "Agile software development: It's about feedback and change". In: *IEEE Computer* 36.6 (2003), pp. 39–43. DOI: 10.1109/MC.2003.1204373.

[67] Christina Wodtke. *Introduction to OKRs*. O'Reilly Media, Inc, 2016.

[68] Robert K. Yin. *Case Study Research: Design and Methods*. 5th ed. Los Angeles, USA: SAGE Publications, 2014.