A Method for Metric Management at a Large-Scale Agile Software Development Organization

Pascal Philipp [1], Franziska Tobisch [1], Leon Menzel [1] and Florian Matthes [1]

[1] *Technical University of Munich (TUM), School of Computation, Information and Technology (CIT), Munich, Germany*

Abstract

The benefits of agile methods for small projects inspired organizations to scale these methods to more extensive settings consisting of multiple agile teams. Such scaling agile settings are more complex, which can make maintaining situational awareness difficult. Metrics can alleviate this problem by increasing insight into the development organization. However, adopting metrics comes with various socio-technical challenges, and current research is missing guidance on metric management in large agile organizations. Therefore, we present a goal-based method designed for a large agile case organization to support stakeholders in selecting, operating, and scaling metrics. Moreover, based on the learnings at the case organization, we present design principles that can potentially guide the development of methods suitable for other contexts. We conducted this research following an action design research (ADR) approach combined with situational method engineering (SME). Our findings indicate that our method proved effective for the case organization. This was accomplished by combining well-established elements (e.g., goal-orientation and tool support) from measurement programs designed for traditional software engineering with unique elements of our method (e.g., metric scaling activities and alignment with agile software development). With this study, we provide deep insights into how metrics are managed at a large agile case organization. Researchers and practitioners can use this work as a foundation for designing measurement programs suitable to other scaling agile organizations.

Keywords

Metrics, Large-scale Agile Software Development, Action Design Research

1. Introduction

Agile methods undeniably succeed when applied in the 'agile sweet spot' characterized by small co-located teams which frequently deliver software and interact with systems with a low to medium criticality and a stable underlying architecture [1, 2]. The demonstrated advantages of agile methods make their application also attractive in larger projects [3]. The relevance of scaling agile software development is evident in practice and research. Numerous organizations scale agile methods by adopting scaling agile frameworks [4, 5]. In parallel, within the scientific community, the number of publications on this topic has increased significantly over the past decade [5].

However, since scaling agile environments are more complex (e.g., multiple teams) than single agile team settings, sustaining an adequate level of communication becomes more difficult. Therefore, to maintain situational awareness, purely relying on qualitative feedback loops is insufficient for effective decision-making. For example, it is hard (i.e., timewise) for a program manager of a large development organization to gain insight by communicating with each agile team regularly. In such circumstances, metrics can augment or replace qualitative feedback.

In traditional software development, metrics are well-researched [6, 7] and heavily used to measure processes, products, and resources' attributes (e.g., costs). Various researchers suggested designs for measurement programs to increase the likelihood of a successful metric adoption [8]. Such measurement programs often rely on or extend the GQM approach [8]. Contrary, in large-scale agile

software development, only a few studies on metrics exist [5] even though researching metrics was declared as a high-priority research topic [9].

A simple strategy to adopt metrics in scaling agile environments could be to appropriate elements from measurement programs designed for traditional software development environments. However, several authors argue against replicating such approaches without modification [10, 11, 12]. For example, a potential risk is that metrics used in traditional software engineering undermine the agile way of working [7, 10]. Generally, embedding metrics in large-scale agile software development organizations presents considerable difficulties, as demonstrated by Philipp et al. [13], who identified eleven challenges (e.g., data collection challenges) that occurred while metrics were adopted in large-scale agile organizations.

However, although introducing metrics in large agile organizations comes with various socio-technical challenges [13], and metrics are widely applied in scaling agile organizations [9], neither scaling agile frameworks nor the scientific literature provide a method to guide practitioners in selecting, operating, and scaling metrics. Therefore, we aim to close this research gap by answering the following research questions (RQs):

RQ1*: How can metrics be selected, operated, and scaled systematically within the scaling agile organization of a large German software company?*

RQ2: *Which design principles can be derived at a large German software company to potentially support the design of future measurement programs at large-scale agile software development organizations?*

We utilized the action design research framework proposed by Sein et al. [14] as our methodological approach to address our research questions. Our investigation leads to the development of a novel method for the goal-based selection, operation, and scaling of metrics, and the formulation of ten design principles with potential applicability across different contexts. The details and implications of these findings will be discussed in the subsequent sections.

The remainder of this paper is structured as follows. In section two, we lay the foundation of this study and provide an overview of related work on metrics in large-scale agile software development. In section three, we introduce the selected research methodology, while in section four, we provide an overview of the research process and present the method we developed for our case organization. In section five, we continue by presenting design principles. After that, we discuss our results and identify future research directions in section six before we conclude this study in section seven.

2. Related Work

Management relies on metrics since they can provide quantitative and concentrated facts to support decision-making [15, 16, 17]. A prominent example of using metrics in organizations is the Balanced Score Card developed by Kaplan et al. [18]. Also in software engineering [17], agile software development (ASD) [19], and large-scale agile software development [20] a widespread application of metrics has been observed.

At first glance, using metrics in ASD and scaling agile development environments seems counterintuitive. For instance, whereas traditional measurement focuses on tracking progress against pre-made plans and measurable goals, embracing change is crucial in ASD [19]. Moreover, various challenges (e.g., data collection challenges, lack of metric usefulness, and metric calculation challenges) add to the difficulty of introducing metrics in scaled agile organizations [13].

Hence, we surveyed the literature on software engineering, ASD, and large-scale agile software development to identify studies that guide metrics selection, operation, or scaling. For software engineering, the systematic literature review by Tahir et al. [8] provide a comprehensive overview of existing measurement planning models and tools, mitigation strategies for their challenges, and success factors in implementing measurement programs.

Tahir et al. [8] identified various approaches for measurement in software engineering, such as Measurement Information Model in ISO/IEC 15939:2007 [21], Goal Question Metrics (GQM) [22, 23,

24], GQM+Strategies [25], Goal Argument Metrics [26] or the Balanced Scorecard [18]. Various models recommended extensions or improvements to the GQM approach [8]. The GQM approach builds upon the assumption that successful measurement must be initiated top-down by identifying the organization's overall and project-related goals [23]. Following goal identification, questions are assigned to goals to characterize the specific object of measurement. Finally, the questions are further refined into metrics which are used to answer the questions quantitatively [27].

Other authors provide a more technical perspective and lack an extensive exploration of the socio-technical environment in which these measurement systems operate. For instance, Staron et al. [28] introduce a framework for designing measurement systems in software development, highlighting the critical role of automated metrics collection and processing in a large organization. Moreover, while the study offers useful insights into the design and operation of measurement systems, it does not fully address the complexities and unique challenges of managing metrics in large-scale agile settings.

In the domain of ASD and large-scale agile software development, several authors use the GQM approach to identify a set of metrics serving different purposes. For example, metrics are selected to build a metrics-based reporting model [29], to establish software process measurement [30], to assess the agile project quality by Kārkliņa et al. [31], to measure the success of agile software development among a set of Critical Success Factors [32], or to measure the impact of an agile transformation [33].

Further, Ram et al. [34] identify success factors for operating metrics in ASD, focusing on factors that facilitate the long-term use of metrics. According to Ram et al. [34], data availability, metric trustworthiness, and development process are particularly important to a successful long-term metrics operation.

An alternative approach to designing, developing, sustaining, and evolving measurement programs is the formation of a dedicated metrics team. In alignment with this perspective, Meding et al. [35] introduced the metrics team maturity model, known as MeTeaM, and provided an assessment template to evaluate the effectiveness of such teams.

In summary, the approaches proposed for ASD focus on supporting metric selection, not long-term metric operation or scaling. Moreover, the proposed approaches for traditional software development were not developed in and adapted to the circumstances of scaling agile environments.

3. Methodology

We chose action design research (ADR) as a design science approach to ensure that the organizational context of the case organization shapes the method developed in this paper. According to Hevner et al. [36], methods can be typical outcomes of design science. The utilization of ADR often demonstrates superiority over traditional design science approaches, for instance due to its capacity to address immediate practical problems [37]. However, it is important to consider the potential drawbacks of this approach, as it typically requires more substantial time commitment and financial resources [37].

During our research, we followed the four stages and seven principles of ADR proposed by Sein et al. [14] (see Figure 1). The first phase, Problem Formulation, identifies and conceptualizes the research problem. During this phase, initial research questions are formulated, and the problem is presented as an instance of a class of problems. Moreover, theoretical foundations and prior technological advances, are identified. This phase contains two principles: Principle 1 declares field problems as opportunities for knowledge creation, and Principle 2 emphasizes that theories inform artifacts created through ADR.

The second phase, Building, Intervention, and Evaluation [7], uses the problem formulated in the first phase and applies the theoretical premises to generate the first draft of the IT artifact. This is further shaped through organizational use and subsequent design cycles. Three principles are characteristic of this phase: Principle 3 emphasizes the mutual influence of artifact and organizational context. Principle 4 highlights the importance of mutual learning among the different project participants. Principle 5 emphasizes the need for authentic and simultaneous evaluation of the artifact in the context of its application.

The third phase is Reflection and Learning. This phase involves reflecting on and documenting what was learned during the intervention and assessment. Principle 6, related to this phase, emphasizes that the artifact reflects not only the preliminary design (c.f., Principle 2) but also the ongoing shaping

through organizational use, perspectives, and participants (c.f., Principles 3 and 4), as well as the results of the authentic, concurrent evaluation (Principle 5).

The fourth phase is the Formalization of Learning. Here, the results are generalized and formalized so that they can be applied to similar situations. Principle 7, associated with this phase, emphasizes the need to generalize the research findings so that they can be applied to other similar situations.

The design principles we outline are not randomly chosen but result from an ongoing ADR process. These principles have been systematically refined and adjusted during each step to effectively reflect the knowledge obtained from our research. Moreover, the principles emerged from a thorough and systematic analysis of our qualitative data.

This process involved transcribing and coding semi-structured interviews, as suggested by Miles et al. [38]. The analytical tool MAXQDA supported our data analysis, helping us delve deeply into the gathered data. We also included insights from numerous workshops and team meetings while deriving these design principles.

In sections four and five, we provide a comprehensive description of the four-stage implementation in this research project, accompanied by the rationale for the principles applied.

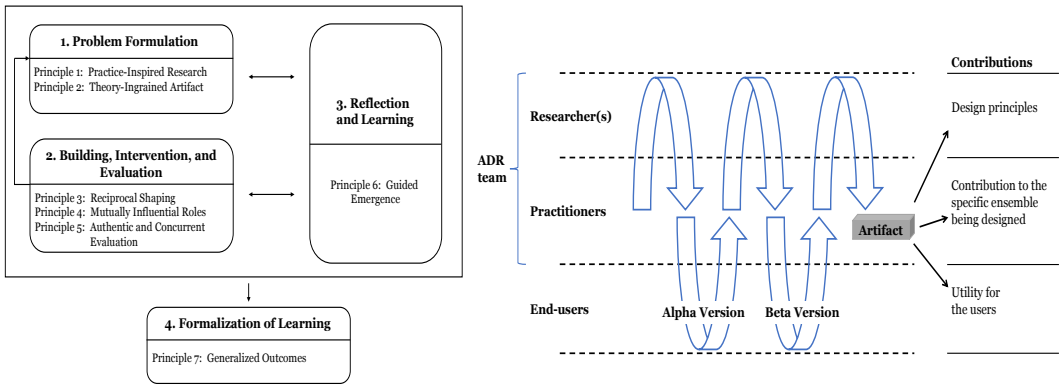

Figure 1: Action Design Research Approach (left) and ADR Stage 2: Building, Intervention, and Evaluation (right) [14]

To design our method in a structured way, we followed the recommendations of Ralyté et al. [39] for conducting situational method engineering (SME). SME effectively complements ADR, since it focuses on project-specific method construction [39]. This approach of integrating SME in ADR also resonates with other studies (e.g., Fridgen et al. [40]).

SME differentiates between method configuration (i.e., adapting a generic method to a specific situation) and method composition (i.e., selecting parts of existing methods that meet the situation's specifics). In this study, we followed the method composition strategy, also known as the assembly-based strategy, since we use elements from existing methods (i.e., GQM Approach) in addition to the elements unique to our method.

The method engineering process consists of three steps. First, we defined the method engineering goal as creating a new method for selecting, operating, and scaling metrics at our large agile case organization. From this goal, we derived the associated requirements. Second, we selected the method chunks that aligned with the requirements. Finally, we assembled these chunks into a new method.

To achieve a robust method design, we incorporated suggested method attributes and elements from Braun et al. [41] and Gutzwiller [42] into our design (see Table 1). To illustrate the method, we use the UML standard for activity diagrams.

4. Results

Below, we describe how we conducted the ADR. First, we outline the class of problems of designing measurement programs in large-scale agile software development, exemplified by the specific problem at the case organizations. After that, the respective building, intervention, and evaluation [7] cycles are described. We started the first BIE stage at InsuranceCo. (March 2021 to September 2021) where we developed an initial design of our method in two organizational units simultaneously. Building on this

initial design, we continued the ADR project at SoftwareCo. (November 2021 to February 2023). Consequently, the final design of our method is rather suited to the context of SoftwareCo. Table 2 provides an overview of both partner organizations.

Table 1
Incorporated Method Attributes and Elements [41, 42]

Fundamental Attributes of Methods
Goal orientation
Systematic approach
Principles
Repeatability
Fundamental Elements of Methods
Role
Activity / Procedure Model
Tool

Table 2
Case Organizations and Involved Roles

Research Partner	Organizational Unit	Description	Partners
InsuranceCo.	IT Provider	The IT Provider is a subsidiary of the InsuranceCo. and employs 30 people in distributed teams to operate and develop software for internal and external customers. The teams use the Scrum-at-Scale framework [43], with sprints lasting 14 days.	Scrum Master Product Owner Principal Product Owner Chief Technology Officer Information Security Officer
	Agile Release Train	The Agile Release Train comprises 80 employees responsible for developing sales software. Since it functions across organizational divisions, it has multiple interfaces with other units. The team uses the Scaled Agile Framework [44] in its Essential configuration.	Scrum Master Developer Release Train Engineer Product Manager
SoftwareCo.	Product Department	The product department consists of more than 40 agile teams working distributed on a large software solution. Within the department, the Scrum of Scrums [45] framework is implemented.	Release Train Engineer Product Manager Developer Scrum Master

4.1. Problem Formulation

We defined the problem to address as "the lack of a systematic method to select, operate, and scale metrics in scaling agile case organizations". We verified this problem and the relevance of a potential solution during focus group discussions and semi-structured interviews with our industry partners InsuranceCo. and SoftwareCo. None of our industry partners had an agreed-upon method to introduce metrics (ADR Principle 1). In both organizations, the few existing metrics were introduced autonomously by single stakeholders or agile teams and were not aligned with the organizational goals.

This problem is also present in other organizations. For example, even though Philipp et al. [13] identified implementing an agile metric management process as the second most mentioned success factor for adopting metrics in large agile software development organizations, none of the practitioner organizations defined such a process. In addition, the industry partners did not have access to expert knowledge, which became apparent across multiple dimensions. For instance, stakeholders within the organizations had limited knowledge regarding metric candidates and the associated challenges of implementing these metrics in scaling agile environments.

To verify the research gap, we reviewed existing literature on large-scale agile software development, which confirmed that previous research did not develop a method to select, operate, and scale metrics in scaling agile environments yet. Moreover, ADR Principle 2 applies to the identified problem since we classify our artifact as theory-ingrained as it builds on the GQM approach [23] and uses the SME approach to guide the design stage.

As suggested by Sein et al. [14] we relate the identified problem at the case organizations to its higher class of problems of measurement programs in large-scale agile software development. This relation is necessary because the goal of ADR is not only to design an artifact that is adapted to the specific conditions of the case organization but also to leverage the learnings obtained at the case organization to derive design principles that apply to a broader class of problems.

Multiple approaches have been proposed to support software organizations in planning their measurement programs [46]. These approaches aim to develop cost-effective measurement programs that only collect useful data with a clear purpose [46]. In essence, these frameworks focus on aligning metrics with organizational goals and the information needs of decision-makers [46].

The suspicion that existing approaches are not explicitly adapted to the requirements of large-scale agile software development organizations served as motivation to form a research program to investigate this problematic situation. For example, it needs to be clarified how measurement programs can be aligned with the development cadence, the agile events, or which agile roles could serve as stakeholders.

4.2. Building, Intervention, and Evaluation

In the second stage, we chose the generic schema for organization-dominant BIE (see Figure 1) to conduct this ADR. Therefore, we deployed the artifact early to challenge organizational participants' existing ideas or assumptions about the artifact's specific use context and to evaluate it iteratively [14]. As suggested by Sein et al. [14], we treated the building of the artifact, its intervention, and evaluation [7] as inseparable and interwoven activities executed concurrently (ADR Principle 3, 5, and 6).

We conducted the BIE cycles (i.e., Alpha and Beta Version) in two target environments at InsuranceCo. and SoftwareCo. (see Table 2). At InsuranceCo., our efforts resulted in an Alpha Version, an early and lightweight design of the method. We based the design of Version Alpha on the GQM approach, as it has proved to be rigorous and adaptable to many different environments [46, 47, 48]. At InsuranceCo. the evaluation sessions were completed during the agile team events, stakeholder discussions, and by a questionnaire.

After that, we built on the Alpha Version at SoftwareCo., where we verified our problem again before we started the next BIE cycle. Our efforts at SoftwareCo. resulted in the eventual and more mature artifact (Beta Version). Hence, the final design of our method is rather suited to the context of SoftwareCo., but it potentially encapsulates some generalizability level since it was developed in two organizations from different industries. At SoftwareCo. we performed the evaluation sessions in group discussions and during the agile team events.

In both target environments, we had the opportunity to interact with practitioners characterized by various software development and management roles (see Table 2). Our research team acted as a sparring partner to the practitioners. The practitioners contributed with their practical hypotheses and knowledge about the organization (ADR Principle 4). In February 2023, we ended the second stage of the ADR at SoftwareCo. since the value of additional cycles seemed marginal.

Figure 2 shows the timeline of our ADR project and the stages conducted at InsurcanceCo. and SoftwareCo. Below, we describe the realized design of our artifact as the outcome of the BIE stage.

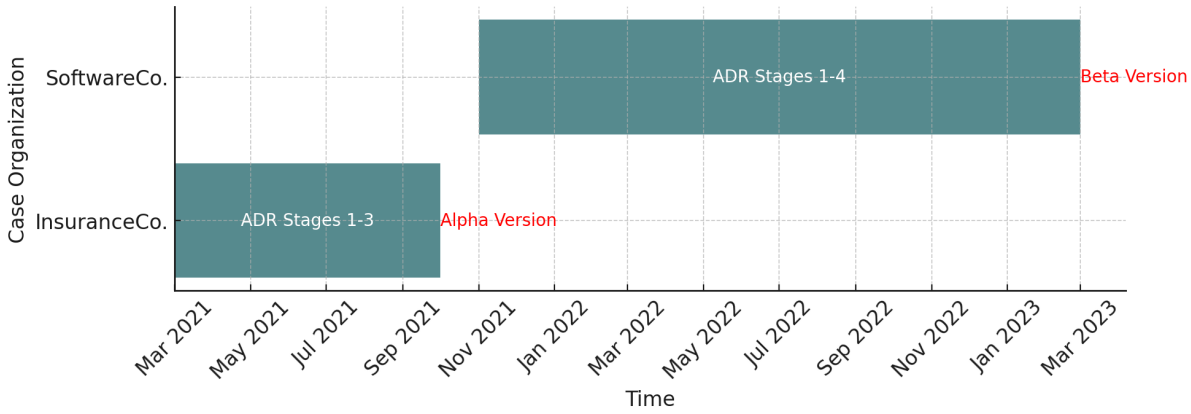

Figure 2: Timeline and Activities of the ADR Project at InsuranceCo. and SoftwareCo.

4.3. Method to Select, Operate, and Scale Metrics in Large-Scale Agile Software Development

To answer our first research question, we have developed a comprehensive method to support the goal-based selection, operation, and scaling of metrics at SoftwareCo. The method is facilitated by the Metric Catalog Web Application and its activities are executed by a metric expert team in collaboration with stakeholders who have or represent an information need. The following will describe the web application, method roles, and each activity.

4.3.1. Metric Catalog Web Application

The Metric Catalog Web Application is not an artifact of this ADR but was developed based on the findings of Philipp et al. [49]. It can be understood as a metric repository consisting of more than 200 predefined metrics. The main functions of the web app are searching for metric candidates and documenting new metrics with their corresponding goals. The metric documentation is done by declaring general metric attributes (e.g., title or calculation rule) and organization-specific metric attributes (e.g., owner and measurement frequency). See Appendix 1 for a detailed screenshot of the Metric Catalog web application user interface.

4.3.2. Roles

The metric expert team and the stakeholders perform the activities of the method together. Both roles are interdependent. The metric expert team requires the stakeholder's knowledge of their agile software development context. In turn, the stakeholders depend on the metric expert team since they usually lack the expertise and resources necessary to adopt metrics efficiently and effectively. Since the method applies to stakeholders with different roles, a top-down (e.g., Domain Managers) and a bottom-up (e.g., Scrum Masters) adoption of metrics is possible. Below, we describe both roles in greater detail.

- ***Stakeholders:*** A stakeholder is an employee at the case organization who has or represents an information need. Metaphorically, the stakeholders can be seen as information customers who

consume the measurements as a service provided by the metric expert team. As soon as the stakeholders can operate the metrics themselves, the metric expert team reduces its involvement and acts as an ad-hoc resource. The primary measure of the method's success is how much value the adopted metrics bring to the stakeholders regarding goal achievement. Stakeholders can be in various agile roles (e.g., Scrum Masters, Product Owners, Program Managers) across different organizational levels (e.g., Team, Program, Portfolio, Organization).

- ***Metric Expert Team:*** The metric team should consist of dedicated experts characterized by long-term availability, a deep understanding of the metric adoption as a socio-technical problem, excellent communication and collaboration skills, and expertise concerning the agile way of working. As a sparring partner to the stakeholders, the metric expert team must deeply understand each method activity and its rationale. Further, the metric expert team intends to increase the value of the method by scaling it within the organization, for instance, through sharing success stories within communities of practice (CoPs), a reoccurring event that invites stakeholders with the same interest to share their knowledge. Finally, for stakeholders with great change resistance, the metric expert team should be able to inspire stakeholders with a standard metric set typical to the role profile of the respective stakeholder. For example, within the case organizations, Scrum Masters tended to be most interested in process metrics, Product Owners and Developers in product metrics, and Program Managers in all metrics (i.e., Product, Resource, and Process). With more than 200 standard metrics, the Metric Catalog Web Application could possibly support the metric expert team in finding a relevant metric set.

4.3.3. Method Main Activities

To improve comprehensibility, we condensed the 20 activities into a simplified diagram (see Figure 3) where they are categorized into the three main activity groups metric selection, operation, and scaling. A comprehensive UML activity diagram illustrating all activities, their interrelationships, and their connections to the Metric Catalog Web Application is provided in Appendix 2.

Metric Selection:

- ***Ensure Management Commitment:*** Management commitment (e.g., Program Manager) is a prerequisite for the initial and every other iteration of the method. Management commitment includes providing necessary resources, such as sponsoring a dedicated metric expert team to serve as a valuable resource for stakeholders throughout the method. Additionally, announcing becoming a metric-driven organization as a goal can motivate stakeholders to fulfill their responsibilities related to the method activities. However, the method is discontinued if a re-evaluation deems it no longer viable and management stops its commitment.

- ***Choose metrics candidates that contribute to goal achievement****:* The connection of metrics to strategic goals ensures that only relevant metrics are introduced. Metrics contribute towards goal achievement by making progress towards a goal more transparent or as a condition to achieve a goal (i.e., metric target values). Goals can be collected from different sources, such as documents, tools, or stakeholders. After goal collection, the gathered goals must be documented in the Metric Catalog Web Application. Next, a measurement priority is determined for each goal. The measurement priority is based on two contributing factors. The first factor, the metric potential, describes the metrics potential to contribute to the achievement of its respective goal, for example by creating transparency on if the goal is likely to be reached or not. The second factor is defined by the goal priority, which describes how significant the goal is for the fulfillment of the organizational strategy. Discussions can serve to estimate the metric potential, goal priority, and measurement priority with the help of Likert values. We consider a detailed estimation procedure superfluous and potentially inefficient since the discussions proved effective at the case organization. Thereafter, following the GQM approach [23], questions and hypotheses are

formulated for the selected goals (i.e., goals with the highest measurement priorities). Finally, the Metric Catalog Web Application is used to search for metric candidates.

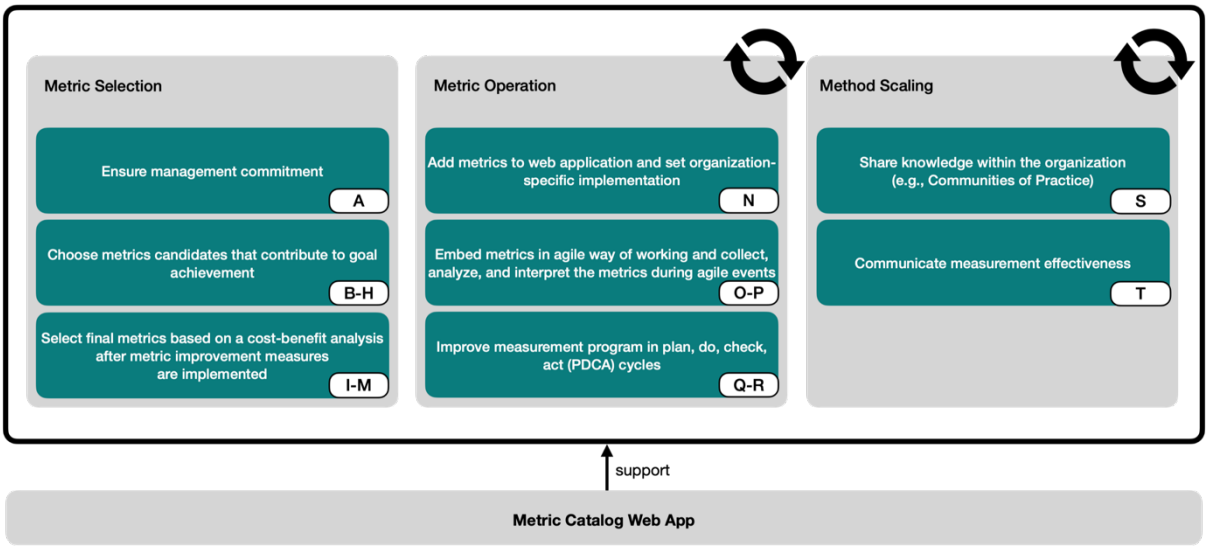

Figure 3: Method to Manage Metrics in Large-Scale Agile Software Development

- ***Select final metrics based on a cost-benefit analysis after metric improvement measures are implemented:*** For each metric candidate, a cost-benefit analysis serves as a decision basis to decide if it is selected. The benefit of a metric is determined by discussing its metric potential. On the cost side, depending on the context of the metric, multiple variables (e.g., data collection, improving data quality) are relevant. To estimate the benefits and costs, again discussions can be leveraged. During these discussions also the potential of cost improvement measures should be determined. However, since each improvement measure also has a cost, it should only be implemented if its potential for metric cost reduction outweighs its implementation cost.

Metric Operation:

The following activities are conducted regularly and aligned with the cadence of the agile events.

- ***Add metrics to the web application and set organization-specific implementation:*** The final set of metrics is added to the Metric Catalog Web Application. Therefore, the corresponding goals are assigned to the metric, and the general and organization-specific elements for each metric are filled out.

- ***Embed metrics in agile way of working and collect, analyze, and interpret the metrics during agile events:*** Metrics should be mapped to the agile event with the most suitable purpose. For example, planning metrics (e.g., velocity) are particularly suitable for the sprint planning event, whereas the sprint retrospective can be a platform to discuss the team happiness metrics. Before each event, the data collection must be finished to enable the analysis and interpretation of the data. Moreover, declaring the metric discussion as a reoccurring event agenda item can facilitate metrics to become an integral part of the agile development process.

- ***Improve measurement program in plan, do, check, act (PDCA) cycles:*** Continuously conducting PDCA cycles to improve the measurement program is necessary to increase its value over time. Important goals for the PDCA cycles could be to increase the quality of individual metrics or to ensure the relevance of the current set of metrics.

Method Scaling:

The following activities are conducted regularly.

- ***Share knowledge within the organization (e.g., Communities of Practice):*** Another option to increase the value of the measurement program lies in scaling it within the organization. Moreover, scaling the method could lead to economies of scale dynamics as introducing new metrics leads to lower long-run average costs (e.g., the metric expert team becomes more efficient). Scaling is accomplished by attracting new stakeholders to participate in the measurement program. An effective tactic to advertise the measurement program lies in sharing how other stakeholders benefited from the metrics (e.g., success stories). Communities of Practice (e.g., Scum Master CoP) could serve as a platform to efficiently communicate to many potential stakeholders.

- ***Communicate measurement effectiveness:*** Reporting on the measurement program's effectiveness serves as a basis for the management to decide if they continue or withdraw their commitment. Therefore, collecting and sharing qualitative and quantitative feedback on how valuable the stakeholders perceive the measurement program should be done regularly.

5. Reflection, Learning, and Formalization

After we ended the second stage, we successfully designed a method to support stakeholders at SoftwareCo. Limited by the nature of ADR, this method is fitted to the context of SoftwareCo. since we based the last design decisions on the context-specific requirements of this target environment. However, since our method is designed to generalize to different stakeholders within SoftwareCo. and relies on objects (e.g., goals and stakeholders) common to most organizations, we suspect some level of generalizability is already included in the current method design. Nevertheless, concerning the Metric Catalog Web Application, it is unclear if other organizations would entrust an externally deployed application with their data. A possible solution to this problem would be to host the application on servers controlled by the respective organization or use the Metric Catalog Web Application for metrics selection, not documentation. Under these considerations, we assume the method can already be applied to the broader class of problems of establishing measurement programs in large-scale agile software development organizations. To answer our second research question, we present ten design principles (ADR Principle 7) below as the output of our continuous reflection on our learnings. These design principles can potentially guide the design of similar methods for other organizations.

- ***Design Principle 1: Ensure management commitment:*** Management must provide the resources necessary to implement the method. Moreover, management should set and communicate becoming a metric-driven organization as a goal to motivate stakeholders to fulfill the responsibilities along the method activities.

- ***Design Principle 2: Establish a metric expert team who views stakeholders as customers:*** The metric expert team should view stakeholders as customers and measurements as a service. Further, the metric expert team should reduce involvement when the stakeholders can operate metrics alone and only act as ad hoc resource.

- ***Design Principle 3: Build stakeholder understanding:*** The metric expert team must ensure that the stakeholders understand the metrics they consume. Stakeholder understanding can be accomplished by involving the stakeholders in setting goals, questions, and hypotheses.

- ***Design Principle 4: Align metrics with agile software development goals****:* Metrics must be aligned with agile software development goals derived from the organizational strategy to ensure alignment.

- ***Design Principle 5: Conduct a cost-benefit analysis to select metrics:*** Metrics with the best cost-benefit ratios should be selected first. On the cost side, we learned that ensuring information quality is a primary driver; on the benefit side, the measurement priority can serve as a decision basis. In addition, the cost-benefit ratio may be enhanced by implementing improvement measures (e.g., improving information quality through defining Jira best practices).

- ***Design Principle 6: Provide a metric repository:*** Finding metric candidates and maintaining the selected set of metrics with a software solution can increase the method's efficiency. Search efficiency can be improved by providing search and filter functionalities. Maintainability is enhanced by typical create, read, update, and delete functionalities for the metrics attributes.

- ***Design Principle 7: Align metric operation with the agile way of working:*** Introducing the measurement program should not lead to losing the benefits of agile methods because the agile way of working is interrupted. In particular, the measurement program must be aligned with the agile way of working (e.g., development cadence, agile events, and agile roles).

- ***Design Principle 8: Maintain and improve the measurement program:*** The measurement program must be continuously maintained and improved. In essence, the quality and relevance of the selected metrics must be ensured continuously. Relevance and quality are critical determinants for the measurement program to provide value to the stakeholders.

- ***Design Principle 9: Scale the measurement program:*** Only maintaining and improving the measurement program limits its value to the current stakeholders. Thus, scaling the measurement program can serve as an option to increase its value since more stakeholders can profit from it. Moreover, scaling the measurement program within the organization can lead to economy of scale dynamics, reducing costs for introducing additional metrics.

- ***Design Principle 10: View the measurement program as an investment:*** The measurement program should only be continued if its future value justifies the investments necessary to run it.

6. Discussion

In response to our first research question, we developed a method that proved effective in supporting stakeholders at SoftwareCo. to manage metrics in their large-scale agile software development organization. Our method focuses on activities to select, operate, and scale metrics. The findings related to SoftwareCo. provide researchers and practitioners with an example of a successful measurement program embedded in a scaling agile environment. Thus, our study contributes to the knowledge of the inner workings of large agile software development organizations.

Moreover, to tackle our second research question, we derived general design principles from our learnings. These principles are relevant for researchers and practitioners since they can potentially help in developing measurement programs in other large-scale agile software development contexts. Moreover, we contribute to the existing body of knowledge by proving that elements of measurement programs designed for traditional software engineering can also be effective in large-scale agile software development. For instance, we build on the GQM approach proposed by Basili et al. [23], include activities to improve the measurement program (i.e., PDCA cycles), prioritize goals, conduct a cost-benefit analysis, and propose setting measurement as a goal.

However, we also contribute to existing research by presenting elements unique to our method. First and foremost, our method aligns the metric operation with the agile software development process. For instance, metrics are mapped to agile events, stakeholders can be of various agile roles, and the PDCA cycles are integrated into the agile software development workflow (c.f., Design Principle 7). Further, unique to our method, the responsibilities of the measurement experts in our method are extended (c.f., Design Principle 9) since they are not only accompanying the metric selection but also the operation and scaling (c.f., Tahir et al. [8] on measurement programs).

Moreover, our method also focuses on scaling the measurement program in the organization (c.f., Design Principle 9), whereas measurement programs designed for traditional software engineering rather focus on sustainability and reusability [8]. Another unique characteristic of our method is determined by the Metric Catalog Web application, which provides a standard set of metrics and corresponding goals for large-scale agile software development (c.f., Design Principle 6).

Concerning the limitations of our study, using ADR inevitably exposes our results to organizational bias. Only two organizations were involved in building our method, and its final design is particularly suited for the environment of SoftwareCo. In addition, each method step may not necessarily be fully developed and likely holds potential for further enhancements. Moreover, the proposed design principles are likely incomplete, and building measurement programs in other target environments would likely result in adding or adapting the design principles.

7. Conclusion

By combining ADR with SME and applying it within two large agile organizations consecutively, we developed a method that guides practitioners through activities to effectively select, operate, and scale metrics within SoftwareCo. This is accomplished by combining elements for measurement programs designed for traditional software development (e.g., Goal-orientation and tool support) with elements unique to our method (e.g., metric scaling activities and alignment with agile software development). Moreover, our method is supported by a metric management team that works closely with stakeholders during the method activities.

In addition, the Metric Catalog Web Application provides support to facilitate metrics selection and operation. Based on our learnings at the case organizations, we provide ten design principles whose applicability may go beyond the case study context. This study is relevant to researchers and practitioners, as the results provide deep insights into an effective measurement program in a large agile organization and can be used to design future measurement programs.

An interesting future research direction is to test the viability of the proposed design principles in other large-scale agile organizations. Moreover, extending the Metric Catalog Web Application to support more activities could result in a more cost-effective measurement program and is, therefore, an interesting future research direction.

8. References

[1] R. L. Nord, I. Ozkaya, and P. Kruchten. Agile in Distress: Architecture to the Rescue. in: Proceedings of the 15th International Conference on Agile Software Development (XP 2014). Rome, Italy. Springer. 2014. p. 43-57. doi: 10.1007/3-540-45017-3_9

[2] P. Kruchten, Contextualizing Agile Software Development, Journal of Software: Evolution and Process. 25.4 (2013) 351-361. doi: 10.1002/smr.572

[3] K. Dikert, M. Paasivaara, and C. Lassenius, Challenges and Success Factors for Large-Scale Agile Transformations: A Systematic Literature Review, Journal of Systems and Software. 119. (2016) 87-108. doi: 10.1016/j.jss.2016.06.013

[4] Digital.ai, 14th Annual State of Agile Report, 2020; URL: https://www.qagile.pl/wp-content/uploads/2020/06/14th-annual-state-of-agile-report.pdf.

[5] Ö. Uludağ, P. Philipp, A. Putta, M. Paasivaara, C. Lassenius, and F. Matthes, Revealing the State of the Art of Large-Scale Agile Development Research: A Systematic Mapping Study, Journal of Systems and Software. 194. (2022). doi: 10.1016/j.jss.2022.111473

[6] N. E. Fenton and M. Neil. Software metrics: roadmap. in: Proceedings of the Conference on the Future of Software Engineering. Limerick, Ireland. ACM. 2000. p. 357-370.

[7] N. Fenton and J. Bieman, Software metrics: a rigorous and practical approach. 3 ed., CRC press, Boca Raton, Florida, USA, 2014.

[8] T. Tahir, G. Rasool, and C. Gencel, A Systematic Literature Review on Software Measurement Programs, Information and Software Technology. 73. (2016) 101-121. doi: 10.1016/j.infsof.2016.01.014

[9] T. Dingsøyr and N. B. Moe. Towards Principles of Large-Scale Agile Development: A Summary of the workshop at XP2014 and a revised research agenda. in: Proceedings of the 15th International Conference on Agile Software Development (XP 2014). Rome, Italy. Springer. 2014. p. 1-8. doi: 10.1007/978-3-319-14358-3
[10] D. Hartmann and R. Dymond. Appropriate agile measurement: using metrics and diagnostics to deliver business value. in: AGILE 2006 (AGILE'06). Minneapolis, Minnesota, USA. IEEE. 2006. p. 126–134. doi: 10.1109/AGILE.2006.17
[11] N. Oza and M. Korkala. Lessons learned in implementing agile software development metrics. in: UK Academy for Information Systems Conference (UKAIS). Oxford, UK. AIS. 2012.
[12] A. Poligadu and R. K. Moloo, An innovative measurement programme for agile governance, International Journal of Agile Systems and Management. 7.1 (2014) 26-60. doi: 10.1504/IJASM.2014.059153
[13] P. Philipp, F. Tobisch, and F. Matthes. Challenges and Success Factors for Metrics in Large-Scale Agile Development. in: Proceedings of the 28th American Conference on Information Systems (AMCIS). Minneapolis, Minnesota, USA. AIS. 2022.
[14] M. K. Sein, O. Henfridsson, S. Purao, M. Rossi, and R. Lindgren, Action Design Research, MIS Quarterly. 35.1 (2011) 37-56. doi: 10.2307/23043488
[15] M. Kütz, Kennzahlen in der IT: Werkzeuge für Controlling und Management. 4 ed., dpunkt. verlag, Heidelberg, 2011.
[16] A. D. Neely, C. Adams, and M. Kennerley, The Performance Prism: The Scorecard for Measuring and Managing Business Success. Prentice Hall Financial Times London, London, 2002.
[17] E. E. Mills. Software Metrics. Carnegie-Mellon University Pittsburgh Software Engineering Institute, Fort Belvoir, Virginia, USA, 1988.
[18] R. S. Kaplan and D. P. Norton, The Balanced Scorecard: Measures that Drive Performance, Harvard Business Review. 70.1 (1992) 71-79.
[19] E. Kupiainen, M. V. Mäntylä, and J. Itkonen, Using Metrics in Agile and Lean Software Development – A Systematic Literature Review of Industrial Studies, Information and Software Technology. 62. (2015) 143-163. doi: 10.1016/j.infsof.2015.02.005
[20] P. Philipp, F. Tobisch, and F. Matthes. Investigating the Adoption of Metrics in Large-Scale Agile Software Development. in: Proceedings of the Pacific Asia Conference on Information Systems (PACIS). Virtual Event. AIS. 2022.
[21] ISO, International Standard - Software Engineering - Software Measurement Process, ISO/IEC 15939, 2002.
[22] V. R. Basili. Software Modeling and Measurement: The Goal/Question/Metric Paradigm. Maryland, USA, 1992.
[23] V. R. Basili, G. Caldiera, and H. D. Rombach, The Goal Question Metric Approach. Encyclopedia of Software Engineering. John Wiley and Sons. 528-532, 1994.
[24] V. R. Basili and D. M. Weiss, A Methodology for Collecting Valid Software Engineering Data, IEEE Transactions on Software Engineering. SE-10.6 (1984) 728–738. doi: 10.1109/TSE.1984.5010301
[25] V. R. Basili, M. Lindvall, M. Regardie, C. Seaman, J. Heidrich, J. Munch, D. Rombach, and A. Trendowicz, Linking Software Development and Business Strategy Through Measurement, IEEE Computer. 43.4 (2010) 57-65. doi: 10.1109/MC.2010.108
[26] Ł. Cyra and J. Górski. Extending GQM by argument structures. in: Proceedings of the 2nd IFIP Central and East European Conference on Software Engineering Techniques. Poznań, Poland. Springer. 2008. p. 26–39. doi: 10.1007/978-3-540-85279-7_3
[27] M. Staron, W. Meding, M. Staron, and W. Meding, *Maintaining and Evolving Measurement Programs*, in *Software Development Measurement Programs: Development, Management and Evolution*. 2018, Springer Cham. p. 225-250.
[28] M. Staron, W. Meding, and C. Nilsson, A framework for developing measurement systems and its industrial evaluation, Information and Software Technology. 51.4 (2009) 721-737. doi: 10.1016/j.infsof.2008.10.001
[29] M. P. Boerman, Z. Lubsen, D. A. Tamburri, and J. Visser. Measuring and Monitoring Agile Development Status. in: Proceedings of the 6th International Workshop on Emerging Trends in Software Metrics. Florence, Italy. IEEE. 2015. p. 54-62. doi: 10.1109/WETSoM.2015.15

[30] P. Ram, P. Rodriguez, and M. Oivo. Software Process Measurement and Related Challenges in Agile Software Development: A Multiple Case Study. in: Proceedings of the 19th International Conference on Product-Focused Software Process Improvement (PROFES 2018). Wolfsburg, Germany. Springer. 2018. p. 272-287. doi: 10.1007/978-3-030-03673-7_20
[31] K. Kārkliņa and R. Pirta, Quality Metrics in Agile Software Development Projects, Information Technology & Management Science. 21. (2018) 54-59. doi: 10.7250/itms-2018-0008
[32] A. M. Aldahmash and A. Gravell. Measuring Success in Agile Software Development Projects: a GQM Approach. in: Proceedings of the 13th International Conference on Software Engineering Advances (ICSEA). Nice, France. IARIA. 2018. p. 38-43.
[33] J. Heidenberg, M. Weijola, K. Mikkonen, and I. Porres. A Metrics Model to Measure the Impact of an Agile Transformation in Large Software Development Organizations. in: Proceedings of the 14th International Conference (XP 2013). Vienna, Austria. Springer. 2013. p. 165-179. doi: 10.1007/978-3-642-38314-4_12
[34] P. Ram, P. Rodriguez, M. Oivo, and S. Martínez-Fernández. Success Factors for Effective Process Metrics Operationalization in Agile Software Development: A Multiple Case Study. in: Proceedings of the International Conference on Software and System Processes (ICSSP 2019). Montreal, Québec, Canada. IEEE. 2019. p. 14-23. doi: 10.1109/ICSSP.2019.00013
[35] W. Meding, M. Staron, and O. Söder, MeTeaM—A method for characterizing mature software metrics teams, Journal of Systems and Software. 180. (2021) 111006. doi: 10.1016/j.jss.2021.111006
[36] A. Hevner, S. March, J. Park, and S. Ram, Design science in information systems research., MIS Quarterly. 28.1 (2004) 75–105.
[37] J. Iivari, Distinguishing and contrasting two strategies for design science research, European Journal of Information Systems. 24. (2015) 107-115. doi: 10.1057/ejis.2013.35
[38] M. B. Miles, A. M. Huberman, and J. Saldaña, Qualitative data analysis: A methods sourcebook. 3 ed., SAGE Publications, Inc., 2014.
[39] J. Ralyté, R. Deneckère, and C. Rolland. Towards a Generic Model for Situational Method Engineering. in: Proceedings of the 15th International Conference on Advanced Information Systems Engineering (CAiSE 2003). Klagenfurt/Velden, Austria. Springer. 2003. p. 95-110. doi: 10.1007/3-540-45017-3_9
[40] G. Fridgen, J. Lockl, S. Radszuwill, A. Rieger, A. Schweizer, and N. Urbach. A Solution in Search of a Problem: A Method for the Development of Blockchain Use Cases. in: Proceedings of the 24th American Conference on Information Systems (AMCIS). New Orleans, Louisiana, USA. AIS. 2018. p. 1-11.
[41] C. Braun, F. Wortmann, M. Hafner, and R. Winter. Method construction-a core approach to organizational engineering. in: Proceedings of the 2005 ACM symposium on Applied computing. 2005. p. 1295-1299. doi: 10.1145/1066677.1066971
[42] T. A. Gutzwiller, Das CC RIM-Referenzmodell für den Entwurf von betrieblichen, transaktionsorientierten Informationssystemen. Vol. 54. Springer-Verlag, 2013.
[43] Scrum Inc., The Official Scrum@Scale Guide, 2021; URL: https://www.scrumatscale.com/scrum-at-scale-guide/.
[44] Scaled Agile Inc., SAFe5 for Lean Enterprises, 2023; URL: https://www.scaledagileframework.com.
[45] K. Schwaber, The Enterprise and Scrum. Microsoft Press, Redmond, Washington, USA, 2007.
[46] C. Gencel, K. Petersen, A. A. Mughal, and M. I. Iqbal, A Decision Support Framework for Metrics Selection in Goal-based Measurement Programs: GQM-DSFMS, Journal of Systems and Software. 86.12 (2013) 3091-3108. doi: 10.1016/j.jss.2013.07.022
[47] P. Berander and P. Jönsson. A Goal Question Metric Based Approach for Efficient Measurement Framework Eefinition. in: Proceedings of the International Symposium on Empirical Software Engineering (ISESE'06). Rio de Janeiro, Brazil. ACM. 2006. p. 316-325. doi: 10.1145/1159733.1159781
[48] A. Boyd, The Goals, Questions, Indicators, Measures (GQIM) Approach to the Measurement of Customer Satisfaction with E-Commerce Web Sites, Aslib Proceedings. 54.3 (2002) 177-187. doi: 10.1108/00012530210441728

[49] P. Philipp, F. Tobisch, L. Menzel, and F. Matthes. Toward a Metric Catalog for Large-Scale Agile Development. in: Proceedings of the 28th American Conference on Information Systems (AMCIS). Minneapolis, Minnesota, USA. AIS. 2022.

Appendix 1

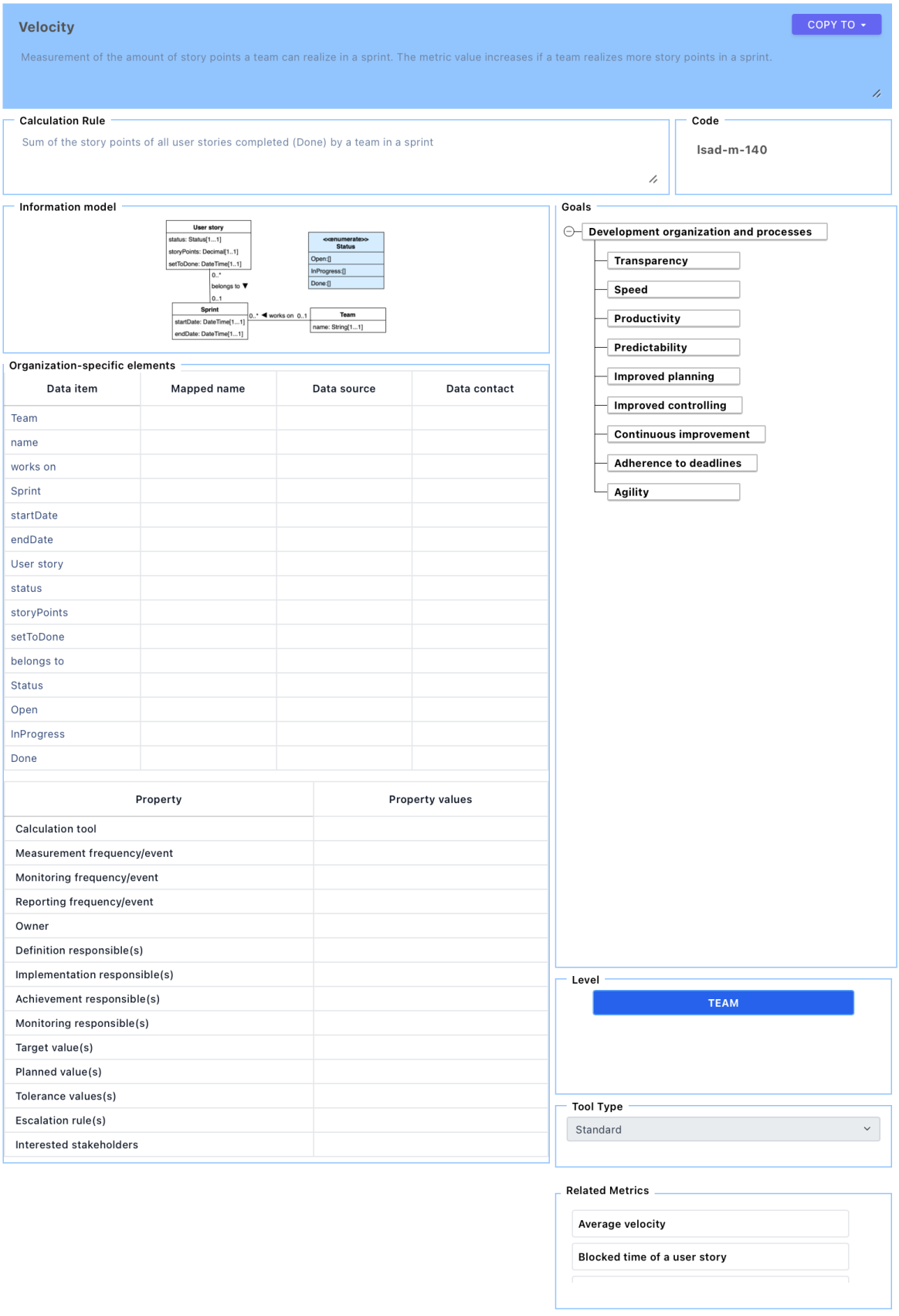

Figure 4: Metric Catalog Web Application

Appendix 2

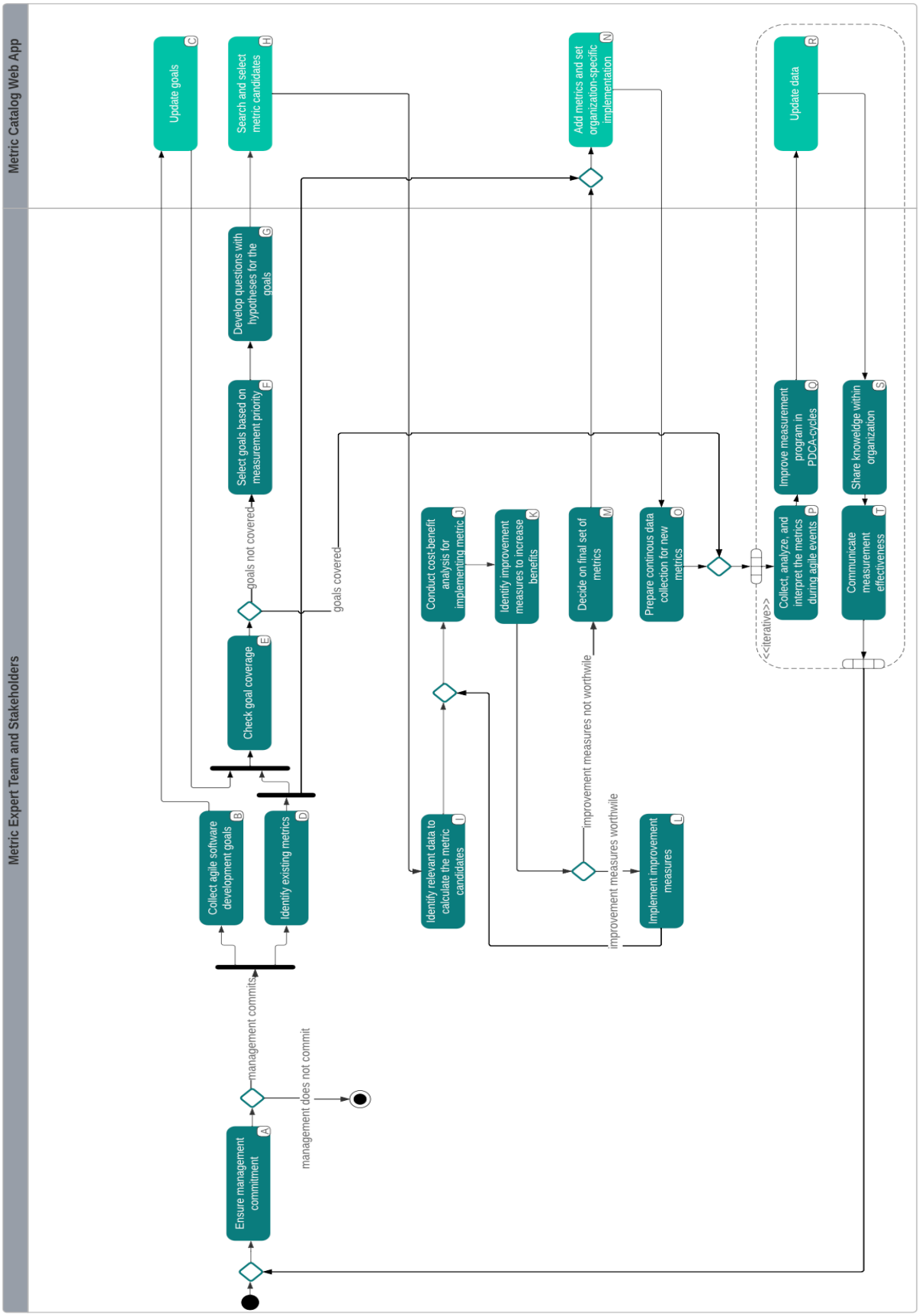

Figure 5: Comprehensive Method Activity Diagram