6-18-2022

Assessing the Impact of Empirical Process Control Metrics in Agile Software Development - A Framework based on Improvement Capability

Tobias Roeck
University of Mannheim, troeck@mail.uni-mannheim.de

Philipp Hoffmann
University of Mannheim, hoffmann@uni-mannheim.de

Kai Spohrer
Frankfurt School of Finance, k.spohrer@fs.de

Tobias Christian Schimmer
University of Mannheim, schimmer@uni-mannheim.de

Armin Heinzl
University of Mannheim, heinzl@uni-mannheim.de

ASSESSING THE IMPACT OF EMPIRICAL PROCESS CONTROL METRICS IN AGILE SOFTWARE DEVELOPMENT - A FRAMEWORK BASED ON IMPROVEMENT CAPABILITY

Research Paper

Röck, Tobias, University of Mannheim, Mannheim, Germany, tobias.roeck@outlook.de

Hoffmann, Philipp, University of Mannheim, Mannheim, Germany, hoffmann@uni-mannheim.de

Spohrer, Kai, Frankfurt School of Finance and Management, Frankfurt, Germany, k.spohrer@fs.de

Schimmer, Tobias, University of Mannheim, Mannheim, Germany, schimmer@uni-mannheim.de

Heinzl, Armin, University of Mannheim, Mannheim, Germany, heinzl@uni-mannheim.de

Abstract

Empirical process control is an integral part of agile software development. A multitude of development metrics has been proposed to implement it. However, the efficacy of control metrics has remained unclear and empirical evidence of their impact is scarce. Methods for assessing whether and how a proposed metric stimulates the improvement of a development process are not yet available. We conduct a design science approach to develop an artifact that assesses the impact of development metrics and we identify their contribution for process improvement at a global software vendor. We draw on the theoretical construct of improvement capability to outline design principles of a measurement framework. Our evaluation of five large-scale agile development projects demonstrates that our framework facilitates to implement development metrics more effectively. The framework has the potential to improve large-scale agile software development and it serves as a useful basis for future empirical research on development metrics.

Keywords: Development Metrics, Empirical Process Control, Improvement Capability, Large-Scale Agile Software Development, Process Improvement,

1 Introduction

Theory on empirical process control and empiricism have been integral parts of agile software development from its first ideas (Schwaber & Sutherland, 2017). Agile software development environments are characterized by small teams, focusing on the planning horizon of a software increment (‘sprint’) in which a defined scope is implemented end-to-end. This incremental and iterative approach aims “to optimize predictability and control risk” (Schwaber & Sutherland, 2017, p. 4). Empiricism assumes that, by gathering data and experience, decisions are made on the basis of what is known, so that empirical process control in agile software development can be defined as the practice of steering and readjusting the agile development processes using emergent, empirical process data rather than designing rigid and static processes or project plans upfront (Schwaber, 1997; Schwaber & Sutherland, 2017). Especially in large-scale agile development systems with a large number of co-located development teams, metrics are used to aggregate information and generate insights to support empirical process control for scheduling, coordination, and other decisions (Bick et al., 2018). In fact, a large variety of distinct software development metrics has been proposed, such as development velocity, cycle time, cumulative flow diagrams, and feature lead time (e.g., Forsgren et al., 2018; Kniberg, 2012).

However, the capability of such metrics to support empirical process control is still unclear and failures to benefit from different metrics have been observed (e.g., Bick et al., 2018). As a result, the large number of choices for metrics provides a barrier to implementing empirical process control in large-scale development contexts. Research offers little guidance on the issue because it lacks approaches to evaluate development metrics. Although some dated research offered approaches to select metrics, such as the 'Goal Question Metric' approach (Basili et al., 1994), prior work lacks empirical evidence on the impact of such metrics. Against this backdrop, this paper aims to answer the following research question:

RQ: How can we design an artifact that assesses the impact of development metrics to ensure that they guide the empirical improvement of software development processes?

Following a design science process as proposed by Peffers et al. (2007), we define an impact measurement framework and iteratively evaluate this framework at a multi-national enterprise software company among 20 agile-working development teams in a large-scale development setting. Our framework relies on improvement capability as a guiding theoretical construct. Building on prior work of Furnival et al. (2017), we define improvement capability as the ability of stakeholders to improve the large-scale software development system they are themselves part of both, intentionally and systematically. We outline design principles that integrate improvement capability into a measurement framework and define its key structural components. Our evaluation based on expert interviews and project management data shows that the framework is useful in understanding the impact of development metrics and in guiding their effective implementation. The framework can be used in future empirical research on development metrics and assists large-scale agile development in practice. Our research offers value for research on empirical process control in agile software development and on software productivity. Likewise, the developed framework artifact can be adopted and implemented by practitioners working with development metrics.

2 Related Work

As part of a systematic literature review, we next provide an overview of related research. The section introduces research on empirical process control to pinpoint the research gap and on the construct of improvement capability as the foundation of our impact measurement framework.

2.1 Empirical Process Control

Research on empirical process control has suggested a multitude of process metrics for software development, such as cycle time, development velocity, and cumulative flow diagrams (e.g., Feyh & Petersen, 2013; Kniberg, 2012). The related research streams in which metrics are proposed include, but are not limited to, software process improvement, value-based software engineering, and software development flow.

First, several frameworks have been proposed to capture development flow. This concept aims to capture how requirements flow through the development process similar to how materials flow through a production process as part of lean manufacturing approaches. Employing varying notations and underlying methods, the objective of these frameworks is to model how items flow through the development process focusing on metrics such as lead time (Ali et al., 2016). Likewise, Petersen and Wohlin (2010) explore avenues to derive continuous improvement opportunities based on measuring the performance of the development process as development flow. Second, a research area adjacent to development flow is concerned with defining and understanding value in software development. Conceptualizing value as value creation points, Mandić et al. (2010) aim to provide a theoretical understanding of value flow in software development. Khurum et al. (2014) present value stream mapping as a method to understand value creating activities in software development and derive basic ratios such as value creating cycle time over total cycle time. Third, besides those examples of academic research, contemporary literature added other framework suggestions such as the FLOW framework by Kersten and Bauer (2018), metrics with a Development Operations (DevOps) focus by Forsgren et al. (2018), and proposals originating from lean software development literature, such as Kniberg (2012).

Yet, the referenced research has not offered insights on how those metrics realize a positive impact towards improvement at the organization. Instead, the suggested measures are merely justified with positive practitioner sentiment obtained from workshops. The referenced examples of contemporary research rely on anecdotal evidence or survey research in the form of experience reports to evaluate their proposals. Hence, academic and popular literature do not provide sufficient rigor to rigorously evaluate the impact of development metrics.

A promising first step to address the research gap is the Goal-Question-Metric (GQM) method by Basili and Weiss (1984) proposed in the 1980s. GQM is a popular methodological approach to define and select metrics. This top-down approach to measurement dates back to early proposals of obtaining performance data for software development (Basili, 1985; Basili & Weiss, 1984). The approach defines a measurement goal as the starting point that is split in several questions. In turn, the questions are split in metrics that can be observed directly (Basili et al., 1994). A modified GQM model is specifically suggested as a guidance model for software organizations to structure measurement programs (Gencel et al., 2013). However, the GQM approach and its mentioned extensions are only concerned with defining and selecting metrics. The approach does not offer means to understand whether the chosen metrics achieve actual improvement in a development system.

To summarize, the existing literature offers rich knowledge on proposing development metrics. However, it does not include the tools and frameworks to evaluate the impact of such metrics on a development system, so that it is still an open question what information is best suited for enabling empirical process control in agile software development teams (Matthies and Hesse, 2019). Existing approaches on defining and selecting suitable metrics are not able to provide an ex-post evaluation.

2.2 Improvement Capability

The construct of improvement capability relates to the dynamic capabilities framework presented by Teece et al. (1997), one of the fundamental frameworks concerned with organizational performance (Furnival et al., 2019). Prior research applied improvement capability in health care contexts examining performance differences of hospitals (Adler et al., 2003). This research found that differences in improvement capability was indeed able to explain performance differences (Adler et al., 2003).

Several definitions of improvement capability can be found in the literature as well as different taxonomies. While Bessant and Francis (1999) capture improvement capability as a dynamic capability defining it as “an organization-wide process of focused and sustained incremental innovation”, other authors focus on human capital and view a “knowledgeable and skilled human resources” as key to improvement capability (Kaminski et al., 2014). Previously referenced Adler et al. (2003), on the other hand, define improvement capability via “resources and processes, supporting both the generation and the diffusion of appropriate innovations”.

Varying definitions of improvement capability result in different taxonomies. Five dimensions which are skills, systems, structures, strategies, and culture make up improvement capability in the resource and processes-focused perspective by Adler et al. (2003). In contrast, five levels ranging from no continuous improvement to the learning organization characterize the learning perspective on improvement capability represented by Bessant and Francis (1999). Furnival et al. (2017) aim to synthesize different conceptualizations and propose eight dimensions. For example, the dimensions include organizational culture and strategy, but also soft factors such as data and performance or employee commitment (Furnival et al., 2017). From a synthesis of existing literature, the authors also propose to define improvement capability as “the organizational ability to intentionally and systematically use improvement approaches, methods and practices, to change processes and products/services to generate improved performance” (Furnival et al., 2017, p. 606). We draw on an adapted definition of Furnival et al. (2017) as the basis to define improvement capability in the context of empirical process control.

3 Methodology

Based on the nature of our research problem, we followed the design science research methodology (DSR) by Peffers et al. (2007). Especially in the context of information systems development (ISD), the IS community aims to contribute to research and practice by developing new artifacts to address and solve unsolved organizational problems (Hevner et al., 2004). We choose the DSR methodology for this research since our research objective is the design of an applicable impact measurement framework (artifact), grounded in existing IS research (rigor), for the unresolved organizational problem of measuring the impact of empirical process control in large-scale development contexts (relevance). We followed the six-staged design science process model, including an iterative refinement of the artifact (Peffers et al., 2007). In total, we conducted four design cycles between the conceptualization, development, instantiation, and evaluation phase.

At the case organization, a set of dashboards showing metrics of prior development increments was implemented with the start of this research project. We conceive the dashboards as part of the empirical context. Importantly, the DSR artifact consists of the impact measurement framework based on improvement capability, not of the dashboards themselves. Our first design cycle was triggered by the case company and initiated with a literature review to identify the research problem. The framework was developed, demonstrated, and evaluated based on an instantiation within the development system of five related new products (Product 1 to Product 5) of a large multi-national enterprise software vendor (see Table 1). We use the term development system to refer to the development organization, its actors, and its empirical context. Three products were already under development and two in the transition from a conceptional pre-development phase to the development phase. The case environment can be characterized as a large-scale agile, cloud native, global ISD setting, including 20 cross-functional development teams in typical “scrum team size” (Schwaber, 1997) and the corresponding management organization. The teams were located in North America (NA), Europe, the Middle East and Africa (EMEA), and Asia-Pacific/Japan (APJ) incorporating different time zones and embedding diverse cultural backgrounds.

Our artifact was first implemented in January 2021 across the 20 ISD teams and productively used ever since. Afterwards, we evaluated and refined our artifact iteratively based on extracted project management system (PMS) data and on 17 expert interviews. The interview partners were chosen from the different teams based on their key roles for identifying and driving improvement topics, incorporating the geographic and hierarchical distribution as well as the different stakeholder groups of the organization. We followed a semi-structured interview approach, where the interviews were scheduled for 30 minutes and recorded with the help of a video conferencing tool. Our interviews were conducted as follows: After a short introduction, the interviewees were first asked what development metrics they used and how they conducted empirical process control in their daily work. In a second step, we discussed the Improvement Capability framework and evaluated their comments. The coding process of the interviews was guided by deductive qualitative content analysis approaches as described in Flick et al. (2010) and Elo and Kyngäs (2008). The coding scheme mapped interview comments to the elements of the framework. When we identified a consistency among the codes from a batch of interviews, we evaluated that consistency as a confirmation for the structure of the framework. While the first design cycle was only guided by the findings from the academic literature, the following three design cycles were mainly informed by the interview feedback resulting in the refinement of the existing artifact design.

We present our research findings from the project in terms of generalized design principles. The design principles are derived from the generalized requirements of our implemented framework. During the design cycles the design principles were refined based on prior research knowledge, the instantiation of the measurement framework of this design cycle including the corresponding design features as well as the iteratively collected feedback from the interviews. Our design principles are similar to 'general components' described by Baskerville and Pries-Heje (2010).

	Product 1	Product 2	Product 3	Product 4	Product 5	Total
Context	large-scale agile enterprise cloud software development					
ISD teams	10	4	2	1	1	20
ISD locations	NA, EMEA	APJ, NA	NA, APJ	NA	NA, EMEA	global
Product phase	dev.	dev.	dev.	concept	concept	
PMS items	5000	2000	1000	-	-	8000
Interview distribution for DSR evaluation						
Scrum masters	3	2	3	1	1	10
Program mgr.	2	2	1	-	-	5
Management	2 cross-product Engineering Manager					2

Table 1. Characteristics of DSR environment

4 Artifact Development of an Impact Measurement Framework

The developed artifact is a measurement framework to understand the impact of development metrics on a development system. The framework provides a tool to assess actors' improvement capability by leveraging development metrics. The following design principles mark the cornerstones of the framework. This section presents the complete impact measurement framework that is the result of our research and the evaluation at a multi-national enterprise software company. We provide detailed information on the process how this artifact was derived from the data and how it was evaluated in the subsequent section.

DP1: Define improvement capability as the guiding construct for impact measurement

Improvement capability serves as the guiding construct to create the impact measurement framework. We define improvement capability as the *ability of stakeholders to improve the large-scale software development system they are themselves part of both, intentionally and systematically*. Our definition is derived from Furnival et al. (2017). We adapt the concept to the setting of a large-scale agile software development system. As part of this adaption, we introduce the notion of stakeholders. For example, the improvement capability of scrum masters determines how capable the scrum masters are in improving the software development system they are involved in. As the purpose of the framework is to measure the impact of using development metrics, "improvement approaches" (Furnival et al., 2017, p. 606) for this framework specifically refers to the use of metrics and dashboards.

The section on related work showed improvement capability as an established theoretical construct. We leverage this construct as a guiding principle for the framework. In the function of a guiding principle, improvement capability defines a shared purpose of development metrics among the stakeholders involved. Thereby, a shared purpose is established not only across the stakeholders involved but also across the pillars introduced by the second design principle. Further, the use of improvement capability as the guiding construct creates a clear definition of success. As identified in the literature review, defining an objective function with the use of metrics constitutes a challenge because value and productivity are hard to grasp in software development (Petersen, 2011). Here, improvement capability offers an alternative to these approaches because it defines a precise, objective function.

Measurement of improvement capability is conducted by evaluating improvement capability as higher, lower or equal compared to a different configuration of metrics usage at a different time. For example, if no development metrics are used at an organization at time *t,* and at time $t + n$ metric dashboards are introduced, the change in improvement capability at time $t + n$ can be measured. Measuring improvement capability numerically is possible as well with the help of quasi-metric scales such as a Likert scale. For example, such numeric measurement can be based on a set of standard questions for interviews. However, the primary purpose of this research is to find a method for impact measurement and to evaluate whether the framework based on improvement capability is useful in that context.

DP2: Include distinct pillars in the measurement structure and assessment independently

For our work, we conceptualize the improvement capability construct with four pillars in sequential order as presented in Figure 2. This taxonomy results from the taxonomies presented by improvement capability literature and drills down one level into the *systems* dimension by (Adler et al., 2003) or *data & performance* by (Furnival et al., 2017), respectively. The composition of the four pillars follows continuous improvement models, e.g., the Deming Cycle (Deming 1982), while the terminology and structure of the pillars is informed by the conceptualization of improvement capability by Bessant and Francis (1999). The four pillars make the construct of improvement capability tangible and actionable in the context of software development process improvement. Each pillar of improvement capability is defined in Table 2. Within each pillar, further measures are defined that detail the impact measurement. From actors' ability to 'Monitor' the process for improvement potential, over the ability to 'Understand' the issue at hand, to the ability to resolving the issue within the 'Act & Improve' pillar, and to 'Control' whether the action achieved the desired effect, the four pillars cover the entire improvement process. The underlying measures of the pillars are conceptualized in Table 2 and design features on their implementation are offered in Table 3. For example, within the 'Monitor' pillar, the two measures active improvements and perceived ability to monitor are brought together to form an overall ability of stakeholders to monitor the performance of the development system.

The analysis of improvement capability overall remains in distinct pillars. As a result, improvement capability is measured as a vector of four dimensions. Each dimension measures the improvement capability of actors in the specific pillar. This allows to understand effects in the four pillars individually. The four dimensions also allow to compare improvement capability between the pillars. As a result, improvement action can be triggered that specifically addresses a certain pillar. It is important to ensure that the chosen pillars offer a both mutually exclusive and collectively exhaustive structure of improvement capability. Such a structure is achieved by introducing the four pillars outlined in Figure 2 that cover the improvement process end-to-end.

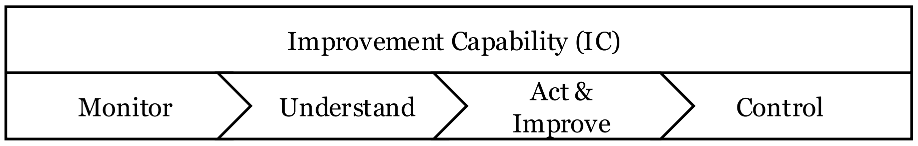

Figure 2. Improvement Capability

DP3: Account for multiple levels of abstraction and stakeholder roles

A large-scale agile development system forms a matrix with different dimensions of complexity. Hence, the impact measurement system needs to account for, and structure said dimensions. Two fundamental dimensions of complexity are the different stakeholder roles and the different products currently in the development system. We structured the product dimension in a team level, a product level, and a multi-product level and map stakeholder roles to each level. The team level includes a single development team that typically consists of fewer than 10 members. A team is working on one or more products at a given time. The product level, also referred to as project or program level, encompasses all activities that relate to one product as a subset of the development system. Team assignment to a product can vary over time. The multi-product level covers multiple products and in turn multiple teams. The primary difference between the three levels is the degree of aggregation for the data shown in dashboards concern with the particular level.

DP4: Combine experience data with structured data

The measures described in Table 2 are based on two primary data sources. First, issue data from a PMS is used for measures such as the existence of duplicate improvements. Second, experience data from interviews informs measures such as actors' perceived ability to monitor the development system. Each pillar includes measures based on one of the two data sources. Consequently, the impact measurement of improvement capability within each pillar is based on both structured issue data and experience data by actors involved in the system.

Pillar	Definition	Measures	Implementation
Monitor	Ability of stakeholders to monitor how well the development system is performing	Active improvements	Project management system items addressing improvements. Open items over time.
		Perceived ability to monitor	Interview answers on perception how the process can be monitored
Understand	Ability of stakeholders to understand the root causes of the current performance	Degree of detail of improvements	Description of project management system items relating to improvements
		Root cause analysis execution by actors	Interview answers on procedure and success in identifying an issue's root cause
Act & Improve	Actions by stakeholders to modify the development system	Actionability of improvements	Completion rate of project management system items that relate to improvements. Existence of child issues for those items.
		Experience with prior improvement actions	Interview answers about improvements initiatives, their effort, and impact
Control	Ability of stakeholders to confirm the positive impact of their actions on the development system	Duplicated improvements	Project management system items that cover the same issue in different products or at different points in time
		Effort to control the impact of improvements	Interview answers on perceived time and effort required to check if improvements achieved positive change

Table 2. Impact Measurement Framework

The structured issue data from the PMS Jira adds an objective perspective to the interviews. The interviews capture soft factors and actor's present sentiment. For specific guidance on how we instantiated the measures, the design features are provided in Table 3.

The Jira data for the measures described in Table 2 can be obtained by exporting it through an integrated Application Programming Interface (API). The data can then be analyzed further in R and Microsoft Excel. Project management data records are structured per issue or ticket and include information in the form of fields such as a 'description' field of the issue or a 'status' field. For example, using the timestamp when the issue was created allows to create a graph that shows how many issues were created per month over the research period as suggested for the measure active improvement.

The data for the interview measures can be obtained by interviewing stakeholders at several points in time. As part of introducing and improving development metrics, actors should be interviewed multiple times to get at least an ex-ante and ex-post understanding. In the interviews, the current setup of metrics can be showed and discussed. Specific interview questions for the interview measures are provided in Table 3.

Data Source	**Pillar**	**Measures**	**Design Features for the Implementation**
Structured	Monitor	Active improvements	Jira data with the issue type 'Improvement'. Number of improvements per month and improvements with the 'Status' field set to 'in progress'
	Understand	Degree of detail of improvements	Jira data with the issue type 'Improvement': Field 'Description' and other text explanations
	Act & Improve	Actionability of improvements	Jira data with the issue type 'Improvement': Field 'Status' to calculate completion rate. Sub-tasks or other child issues created for improvements
	Control	Duplicated improvements	Jira data with the issue type 'Improvement' that address the same underlying issue in sequence
Experience-based	Monitor	Perceived ability to monitor	Interview answers to the following questions: How is the team / project doing in terms of development flow? Why do you think so? Did you look at any data about the development process? What was the last issue you encountered with your team / project?
	Understand	Root cause analysis execution by actors	Interview answers to the following questions: How did you find out the root cause of the last impediment your team / project encountered? You find that velocity is going down for your team / project. What do you do? What information about the team / project do you wish for?
	Act & Improve	Experience with prior improvement actions	Interview answers to the following questions: How did you proceed when you find out the root cause of an issue? What do you wish for with regards to data about the refinement & development process?
	Control	Effort to control the impact of improvements	Interview answers to the following questions: Did the action achieve the desired outcome? How did you check for the desired outcome?

Table 3. Design Features of the Impact Measurement Framework

To instantiate the framework in a large-scale agile development setting the design features need to be aligned with the specific development system. However, it should be ensured that the framework presented in Table 2 including its design principles act as the blueprint for such an instantiation. Afterwards, the required data must be collected (see Table 3). Building on the collected data, the improvement capability of actors is assessed within the four pillars of the framework.

5 Evaluation

The framework presented in the section above is the result of four design cycles in which the measures and their definition were refined for practical relevance and conceptual soundness.

DP1: Defining improvement capability as the guiding construct communicates the purpose of using development metrics clearly. Improvement capability also captures the fundamental raison d'être for using development metrics in the first place. Specifically, dashboards of development metrics enable actors to improve the processes they are involved in. This point is supported by actors' perspective on development metrics and their presumed reason for using development metrics. In the evaluation interviews, two scrum masters explained that dashboards allow them to improve the sprint planning process and allow them to better coordinate with the team.

"We like that [table] there because we normally keep changing in real time when we're doing our planning sessions to know who has not been assigned anything or who has been overload and now, we can be in a position to balance the assignment of tasks." (Scrum master, Product 3)

"It shows me the people that don't log their work. Not that I want to track them, but that's a pain point we've had." (Scrum master, Product 1)

The presented quotes demonstrate that development metrics enable actors to improve the development system. As the dashboards enable the scrum masters to improve the planning session, they are capable of improving the development system overall. Since improvement is the primary reason to use dashboards, improvement capability provides the guiding construct for the impact measurement framework.

In addition, the guiding construct streamlines the organization of metrics tracked for different purposes. For example, quality metrics such as automated test coverage and a number of security metrics are tracked at the case organization as well. Those metrics are analyzed in a developer portal implemented inhouse. Although some of those metrics could be relevant for improvement as well, so far, they are not integrated with data from the project management system. Improvement capability as a construct supports the classification of metrics and it ensures the focus on only relevant metrics for process improvement. For example, security metrics relating to a given product may have no direct impact on process improvement and should therefore not be included in a process dashboard but considered in a product or release view.

Thus, the guiding construct to measure the impact of development metrics gives a clear purpose of using these metrics. In addition, improvement capability facilitates organizing a potentially large number of tracked metrics by their purpose.

DP2: The second design principle ensures a precise and end-to-end measurement. This is achieved by including the pillars 'Monitor', 'Understand', 'Act & Improve', and 'Control'. This design principle is based on the observation that improvement capability as a result of using metrics indeed differs between the defined pillars.

This point becomes apparent from the following example: The dashboards in place achieve strong improvement capability in pillar four, 'Control'. Particularly, the analyzed improvement items show that, prior to the dashboards, there were many improvements that addressed the same underlying issue again and again at different points in time. One frequent problem concerns work items that were scheduled for implementation in a sprint but were not ready and required further work by product managers or designers. This problem was raised as an improvement item six times between October 2019 and March 2021.

A control mechanism was implemented for the problem on a dashboard. A filtered overview of issues was added that shows issues scheduled for the upcoming increment. In addition, a checklist for items scheduled for implementation was introduced. Remarkably, the action was driven proactively by one program manager.

"I need a way to check if [work items] fulfill the ready criteria before scheduling them for a sprint. I created a dashboard element myself to achieve this." (Program manager, Product 1)

The example indicates how dashboards raised the improvement capability of actors in the 'Control' pillar. In contrast, the effect of the dashboards on actors' ability to 'Act & Improve' was not as substantial. An effect merely occurs when materializing initial benefits of dashboards results in a virtuous cycle of increasing effectiveness of improvements leading to higher motivation to keep the underlying dashboard data up to date which in turn leads to better actions derived from the dashboards. The following quote illustrates the start of such a cycle.

"Okay, so this [work balance chart] will be useful in the longer run, because right now we are still transitioning. […] Starting probably either next sprint or the one after […], but that's why it looks like that right now." (Scrum master, Product 1)

Because the framework measures improvement capability separately for the pillars, such differences in improvement capability can be addressed by modifying development metrics further. For example, other metrics may support improvement capability for the 'Act & Improve' pillar.

The ability to measure differences in improvement capability for different pillars underlines the importance of the second design principle. Therefore, impact measurement in terms of improvement capability requires to be distinguished in distinct pillars end-to-end. As a result, the impact of development metrics can also be evaluated in comparison across different pillars.

DP3: The third design principle introduces multiple layers of abstraction in the impact measurement framework. This is important because actors show a tendency to focus strictly on their own role perspective. For example, interviews with scrum masters on a team level revealed that they were skeptical towards metrics that could be used for projections, i.e. forecasts of when a certain work item was expected to be completed. The scrum masters justified their skepticism with prior experiences with management exerting pressure on the teams to deliver on inadequate projections.

"I'm a pretty skeptic person naturally, and the team has been in a difficult position in the past because of really, really poor-quality projections, and I just really don't want that to happen again." (Scrum master, Product 1)

However, on the level of an engineering manager, projections become the key use case of development metrics as disclosed by the following quote.

"Out of a couple of KPIs, I think what is important is predictability so that we can sort of get a good assessment for when we may be able to deliver something." (Engineering manager)

The quotes express that accounting for multiple levels of abstraction and distinguishing three levels from team to multi-product level also helps to resolve seemingly conflicting impacts of metrics. As shown above, the impact of the same metrics can be different on different levels of abstraction. For example, the impact of metrics that allow projections can be understood separately for the multi-product level and for the team level as seen above. Thus, including multiple levels of abstraction in the framework is necessary for a complete understanding of the effects of metrics.

DP4: The fourth design principle proposes to include both experience data and structured data. The two data sources are found to complement each other. On the one hand, the inclusion of measures that capture actors' experience made it possible to uncover unknown unknowns about the used dashboards. On the other hand, the measures based on structured data from a project management system provided timestamped information about the development system before the dashboards were introduced. Both data sources have their advantages and disadvantages:

First, an advantage of interviews is the ability to uncover unknown unknowns. An example for this is the choice of aggregation method. Most development metrics require a sum of work per increment or person. For example, work items are counted or a size estimate such as story points is accumulated. Alternatively, the use of time estimates for the expected duration of work is discussed. Ultimately, both the usefulness and the relevance of the dashboards could be improved by uncovering the importance of deciding on an aggregation method. A disadvantage of interviews is the susceptibility to volatile and potentially misguided sentiment of the interviewees. For example, actors frequently underestimated the size of issues. With regards to the impact measurement framework as a whole, including issue data limits the dependency of impact measurement on the current sentiment within the development system by adding data from a longer timeframe. As shown, experience data and structured data complement each other to form a more complete picture of improvement capability. Hence, both types of data should be included in the framework.

The evaluation demonstrates the value of the framework in assessing the impact of development metrics. The importance of improvement capability as a guiding construct and the benefits of measuring improvement capability in distinct pillars are only two examples of critical design principles. It is concluded that the framework as a whole and its design principles are useful in understanding the impact of development metrics on a development system.

6 Discussion

This paper has suggested an impact measurement framework for development metrics. The framework offers a comprehensive understanding of the impact development metrics have on the continuous improvement of development systems. The paper presents design principles for impact measurement frameworks and evaluates the suggested framework in a large-scale agile setting at a multi-national enterprise software organization. The evaluation indicates the value of the design principles as well as the overall usefulness of the framework in understanding the impact of development metrics on a development system. The paper fills the research gap of examining the effects of empirical process controls and it provides the conceptual means to implement empirical process control. As described in Matthies and Hesse (2019) and Fitzgerald et al. (2014), there are few details available that specify how to conduct empirical process control specifically. Indeed, the authors find "a pressing need to better understand […] lean concepts" (Fitzgerald et al., 2014, p. 93). With this paper, we contribute to gaining such an understanding from both the perspective of a researcher and that of a practitioner.

For research, both the impact measurement framework itself and insights revealed by applying the framework provide valuable contributions. First, this work adds to prior research efforts by adopting improvement capability as an underlying theoretical construct for an improvements measurement framework (Furnival et al., 2017). This allows the framework to obtain a purpose-driven and integrated view on the development system. Previous research on informed decision making in software development environments often defined a single use goal within a goal-question-metric procedure as suggested in Gencel et al. (2013). However, such goals are only applicable temporarily and do not cover the entire development system. In contrast, continuous improvement can serve as a persistent purpose of using development metrics. Therefore, improvement capability should be the underlying objective of future research endeavors regarding software process improvement.

Second, the framework complements metric suggestions on development flow and value-based software engineering. This is achieved as the framework provides the tool to evaluate the impact of metrics such as cumulative flow diagrams or value creating cycle time (Khurum et al., 2014; Petersen & Wohlin, 2011). Such prior research suggested a large pool of development metrics but did not provide sufficient evidence for their usefulness. The suggested framework allows now to evaluate and justify future metric proposals in terms of their improvement capability. By implementing the framework, future research should include such a performance evaluation of metrics.

Third, the impact measurement framework serves as an enabler to create structured data on the impact of development metrics on a development system. The components and measures of the framework are clearly defined and advice on its implementation with project management systems is provided. Hence, research and practice can readily adopt the framework. As a result, the framework streamlines modelling efforts for development flow such as Ali et al. (2016). The suggested models of item flow can now be benchmarked in terms of their capability to achieve improvement at the development system they are used in. Consequently, different models for item flow can now be compared with the framework.

For practitioners, the impact measurement framework constitutes a tool for implementing empirical process control. Specifically, measuring improvement capability with the framework and subsequently adapting the metrics in place resulted in greater acceptance and higher usability of the dashboards. In addition, the emphasis on change management activities when rolling out development metrics is an important insight to forestall adverse reactions to transparency. Employing improvement capability as the guiding construct streamlines the communication of a clear purpose of metrics. Measuring the impact of metrics with the presented framework also allows to focus change management activities on those specific pillars of improvement capability that rank lower than others.

As stated above, the framework enables a structured collection of data on the impact of metrics. Future work should collect such data in different organizational contexts. This enables research to theoretically understand the impact of development metrics on the development system and also identify relevant contingency factors and moderators of the observed impact. This line of inquiry is important to develop a comprehensive construct for implementing empirical process control as called for by Matthies and Hesse (2019) and Fitzgerald et al. (2014).

Future research may also extend the taxonomy of improvement capability and reassess its pillars and underlying measures. Existing research on improvement capability such as Bessant and Francis (1999) and Adler et al. (2003) outlines additional dimensions that may be included in the framework. For example, a potential additional dimension is process improvement and learning (Furnival et al., 2019). Future research should analyze whether including additional dimensions of improvement capability improves the framework further.

A limitation of this work results from the restricted timeframe of the research period. In hindsight a longer research period would have allowed to observe ex-ante and ex-post effects with the use of the impact measurement framework in more detail. However, we plan to continue our observations and will iterate through the lifecycles of several improvements after the introduction of the impact measurement framework. In addition, the external validity of the framework in other contexts poses another limitation. We evaluated the framework at a multi-national enterprise software company within a large-scale agile development setting that followed the Scrum methodology. However, we cannot directly infer that the framework is applicable in other agile settings as well.

7 Conclusion

This paper presented and evaluated an impact measurement framework for software development metrics. By incorporating improvement capability as its guiding construct, the framework offers an integrated view on the development system. Four design principles outline key structural characteristics of the framework such as including distinct pillars of measurement and distinguishing multiple levels of abstraction. The evaluation showed that the framework is valuable in understanding the impact of metrics. The framework also has implications for future research. Namely, research should benchmark suggested development metrics with improvement capability. Building on the presented framework, future research should develop an understanding of the determinants and contingency factors that influence the impact of development metrics on the development system.

References

Adler, P. S., Riley, P., Kwon, S.-W., Signer, J., Lee, B., & Satrasala, R. (2003). Performance Improvement Capability: Keys to Accelerating Performance Improvement in Hospitals. California Management Review, 45(2), 12–33. https://doi.org/10.2307/41166163

Ali, N. B., Petersen, K., & Schneider, K. (2016). FLOW-assisted value stream mapping in the early phases of large-scale software development. Journal of Systems and Software, 111, 213–227. https://doi.org/10.1016/j.jss.2015.10.013

Basili, V. R. (1985). Quantitative evaluation of software methodology (TR-1519). University of Maryland.

Basili, V. R., Caldiera, G., & Rombach, H. D. (1994). The Goal Question Metric Approach. Encyclopedia of Software Engineering.

Basili, V. R., & Weiss, D. M. (1984). A Methodology for Collecting Valid Software Engineering Data. IEEE Transactions on Software Engineering, 10(6), 728–738. https://doi.org/10.1109/TSE.1984.5010301

Baskerville, R., & Pries-Heje, J. (2010). Explanatory Design Theory. Business & Information Systems Engineering, 2(5), 271–282. https://doi.org/10.1007/s12599-010-0118-4

Bick, S., Spohrer, K., Hoda, R., Scheerer, A., & Heinzl, A. (2018). Coordination Challenges in Large-Scale Software Development: A Case Study of Planning Misalignment in Hybrid Settings. IEEE Transactions on Software Engineering, 44(10), 932–950. https://doi.org/10.1109/TSE.2017.2730870

Bessant, J., & Francis, D. (1999). Developing strategic continuous improvement capability. International Journal of Operations & Production Management, 19(11), 1106–1119. https://doi.org/10.1108/01443579910291032

Deming, E. W. (1982). Out of crisis, Cambridge, Mass. Massachusetts Institute of Technology.

Elo, S., & Kyngäs, H. (2008). The qualitative content analysis process. Journal of Advanced Nursing, 62(1), 107–115. https://doi.org/10.1111/j.1365-2648.2007.04569.x

Feyh, M., & Petersen, K. (2013). Lean Software Development Measures and Indicators - A Systematic Mapping Study. In B. Fitzgerald, K. Conboy, K. Power, R. Valerdi, L. Morgan, & K.-J. Stol (Eds.), Lecture Notes in Business Information Processing: Vol. 167. Lean Enterprise Software and Systems: 4th International Conference, LESS 2013, Galway, Ireland, December 1-4, 2013, Proceedings (Vol. 167, pp. 32–47). Springer. https://doi.org/10.1007/978-3-642-44930-7_3

Fitzgerald, B., Musiał, M., & Stol, K. J. (2014). Evidence-based decision making in lean software project management. In Companion proceedings of the 36th international conference on software engineering, 93-102. https://doi.org/10.1145/2591062.2591190

Flick, U., Kardorff, E. von, & Steinke, I. (2010). A companion to qualitative research (Repr). SAGE.

Forsgren, N., Humble, J., & Kim, G. (2018). Accelerate: Building and scaling high performing technology organizations (First edition). IT Revolution Press. 4085

Furnival, J., Boaden, R., & Walshe, K. (2017). Conceptualizing and assessing improvement capability: A review. International Journal for Quality in Health Care, 29(5), 604–611. https://doi.org/10.1093/intqhc/mzx088

Furnival, J., Boaden, R., & Walshe, K. (2019). A dynamic capabilities view of improvement capability. Journal of Health Organization and Management, 33(7/8), 821–834. https://doi.org/10.1108/JHOM-11-2018-0342

Gencel, C., Petersen, K., Mughal, A. A., & Iqbal, M. I. (2013). A decision support framework for metrics selection in goal-based measurement programs: GQM-DSFMS. Journal of Systems and Software, 86(12), 3091–3108. https://doi.org/10.1016/j.jss.2013.07.022

Hevner, March, Park, & Ram (2004). Design Science in Information Systems Research. MIS Quarterly, 28(1), 75. https://doi.org/10.2307/25148625

Kaminski, G. M., Schoettker, P. J., Alessandrini, E. A., Luzader, C., & Kotagal, U. (2014). A comprehensive model to build improvement capability in a pediatric academic medical center. Academic Pediatrics, 14(1), 29–39. https://doi.org/10.1016/j.acap.2013.02.007

Kersten, M., & Bauer, E. (2018). Project to product: How to survive and thrive in the age of digital disruption with the flow framework (G. Kim, Ed.) (First edition). IT Revolution Press.

Khurum, M., Petersen, K., & Gorschek, T. (2014). Extending value stream mapping through waste definition beyond customer perspective. Journal of Software: Evolution and Process, 26(12), 1074–1105. https://doi.org/10.1002/smr.1647

Kniberg, H. (2012). Lean from the Trenches: Managing large-scale projects with Kanban. The pragmatic programmers. O'Reilly Media.

Mandić, V., Oivo, M., Rodríguez, P., Kuvaja, P., Kaikkonen, H., & Turhan, B. (2010). What Is Flowing in Lean Software Development? In P. Abrahamsson (Ed.), Lecture Notes in Business Information

Processing: Vol. 65, Lean Enterprise Software and Systems: First International Conference, LESS 2010, Helsinki, Finland, October 17-20, 2010. Proceedings (pp. 72–84). Springer Berlin Heidelberg.

Matthies, C., & Hesse, G. (2019). Towards using data to inform decisions in agile software development: views of available data. arXiv preprint arXiv:1907.12959. https://doi.org/10.48550/arXiv.1907.12959

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A Design Science Research Methodology for Information Systems Research. Journal of Management Information Systems, 24(3), 45–77. https://doi.org/10.2753/MIS0742-1222240302

Petersen, K. (2011). Measuring and predicting software productivity: A systematic map and review. Information and Software Technology, 53(4), 317–343. https://doi.org/10.1016/j.infsof.2010.12.001

Petersen, K., & Wohlin, C. (2010). Software process improvement through the Lean Measurement (SPI-LEAM) method. Journal of Systems and Software, 83(7), 1275–1287. https://doi.org/10.1016/j.jss.2010.02.005

Petersen, K., & Wohlin, C. (2011). Measuring the flow in lean software development. Software: Practice and Experience, 41(9), 975–996. https://doi.org/10.1002/spe.975

Schwaber, K. (1997). SCRUM Development Process. In J. V. Sutherland (Ed.), Business object design and implementation: Oopsla '95 workshop proceedings 16 October, 1995, Austin Texas (pp. 117–134). Springer. https://doi.org/10.1007/978-1-4471-0947-1_11

Schwaber, K., & Sutherland, J. (2017, November 1). The Scrum Guide. scrum.org. https://www.scrumguides.org/scrum-guide.html

Teece, D. J., Pisano, G., & Shuen, A. (1997). Dynamic capabilities and strategic management. Strategic Management Journal, 18(7), 509–533.