Received: 11 December 2020 | Revised: 8 October 2021 | Accepted: 29 October 2021

IET Software

DOI: 10.1049/sfw2.12051

The Institution of Engineering and Technology WILEY

ORIGINAL RESEARCH PAPER

Value-oriented quality metrics in software development: Practical relevance from a software engineering perspective

Philipp Haindl[1,2] | Reinhold Plösch[1]

[1]Department of Business Informatics - Software Engineering, Johannes Kepler University, Linz, Austria

[2]Software Competence Center Hagenberg, Hagenberg, Austria

Correspondence

Philipp Haindl, Software Competence Center Hagenberg, Hagenberg, Austria.
Email: philipp.haindl@jku.at and philipp.haindl@scch.at

Abstract

When following the principles of value-based software engineering, business, customer satisfaction, and engineering considerations need to be balanced to develop and operate the software so that it satisfies the different stakeholders' expectations. This, however, requires knowing the relevant quality metrics covering these value-oriented expectations and potential sources for their measurement. In this work, a categorisation of value-oriented quality metrics that are practically relevant is presented. Therefore, the authors conducted an online survey with practitioners who assessed the relevance of 61 value-oriented metrics, gathered from a preceding systematic mapping study. The authors grouped these metrics into 10 categories, based on financial, customer satisfaction, value proposition, and creation perspectives. Also, the authors examined the frequency of particular steps at which these measures accrue and identified their most relevant data sources. The participants rated metrics for feature reliability, performance, as well as test and development efficiency as most relevant for value orientation. According to the participants, the authors' collection covers all relevant metrics for addressing financial and market and feature usability aspects. The authors' categorisation and the metrics' relevance assessments shall support software engineers in selecting relevant metrics and their sources for software product development.

KEYWORDS

software development management, software engineering, software metrics, software quality

1 | INTRODUCTION

Value-based software engineering allows regarding different stakeholders' value appraisal of the developed software by evaluating development and operational metrics relevant for the stakeholders. Thereby, these metrics serve as indicators of how well specific values for the respective stakeholder are fulfilled. Appropriate quality metrics for value-based software engineering must take into account the different value considerations of stakeholders, for example, business, security, customer-related or technical considerations [1, 2].

However, existing classifications of quality metrics isolatedly focus on one single-stakeholder perspective, such as the development perspective [3–5], the business perspective for the customer value analysis [6, 7], the customer perspective [8, 9], or the perspective of feature ideation and improvement [10, 11]. The common notion of software quality metrics primarily captures quality-related aspects that contribute value for technical stakeholders, for example, maintainability or performance efficiency of the software. As such, they neglect aspects that are particularly relevant for the value appraisal of stakeholders outside the technical domain, for example, for business stakeholders who expect that the software is frequently used by customers or generates revenue aligned with the business model. As an example, after deploying a new feature of a mobile app into production, observing changes of the customers' satisfaction in app stores, app outage reports, or customer churn is paramount for deciding about required operational adjustments, bug fixes, or code improvements in subsequent development phases. However, due to the absence of a comprehensive list and classification of metrics addressing the different stakeholders' value-oriented considerations, it is cumbersome to meet these value expectations.

This paper seeks to close this research gap by presenting a concrete list and categorisation of practically relevant value-oriented quality metrics. The metrics are based on our former systematic mapping study of over 140 papers [12] and a complementary narrative literature review. The relevance and completeness of these metrics were assessed based on the results from a survey with 41 practitioners who have different roles in the software product lifecycle. Also, we present the identified sources in companies for measuring these metrics in practice.

The paper is divided into 10 sections, whereby the remainder of this section elaborates the research objective and contributions of this study. Section 3 describes the research methodology and particularly the questionnaire design and participant recruitment for the survey. Section 4 details the demographic aspects, professional information, and backgrounds of the participants. Following, Section 5 highlights our findings regarding the software development practice in companies. The relevance assessments of the metrics through the participants are discussed in Section 6, complemented by the categorisation of the metrics along value-oriented stakeholder perspectives [13] presented in Section 7. The results from assessing the completeness of the metrics are given in Section 8 whereby we also elaborate on the metrics declared as missing by the participants. We summarised these missing metrics through qualitatively analysing individual survey responses. In Section 9, we discuss the applicability of the results from the perspective of practitioners and researchers, respectively. Section 10 outlines the recognised threats to the validity of the presented results, before concluding our work and sketching possible directions for future work in Section 11.

1.1 | Research goal and contributions

The overall research goal of this paper was on gathering empirical data regarding the relevance and completeness of quality metrics for selected value considerations, identified through the preceding systematic mapping study. This research goal can be refined into the following objectives:

- Assessing the practical relevance of the metrics identified by our literature study, from the perspective of practitioners in the software product lifecycle.
- Evaluating to which degree a metric category is covered by the associated metrics.
- Detecting additional metrics having value for critical stakeholders in software development.
- Identifying suggested sources for measuring these metrics.

When applying the idea of value-based software engineering [14], different value considerations from business, customer, and development stakeholders need to be addressed and specified in a concrete and unambiguous manner using quantitative quality metrics. These metrics also facilitate balancing contradictory value considerations of stakeholders and evaluating them in relation to a company's business model [15–17].

The presented paper contributes to the current research challenges for value-oriented software engineering as follows:

- A categorisation of value-oriented quality metrics, whose necessity was identified through empirical case studies and the lack of the same criticised by several authors [5, 18, 19].
- Empirical findings regarding the practical relevance of value-oriented quality metrics from the perspectives of software engineering practitioners. The authors of Refs. [15, 17], thereby, especially highlighted roles such as product and project management, software development, and operation management as important.
- A summary of the most relevant measurement sources to operationalise these metrics in practice. Knowing these measurement sources is the prerequisite to developing appropriate software tools [20, 21] that automatically acquire and evaluate these metrics.

2 | RELATED WORK

In a qualitative multiple-case study [18], practitioners were interviewed about the practical challenges of DevOps adoption in companies. The respondents mentioned that common classifications of software quality metrics primarily focus on software product quality, thus neglecting metrics focussing on the process of software development, delivery, and operation themselves. Similarly, the authors in Ref. [19] conducted an exploratory case study interviewing several software engineers about quality-related requirements in DevOps. The interviews unveiled that in their projects the experts often implicitly are confronted with non-technical requirements such as, for example, the alignment of the software with the business model, regulatory considerations or the demand for user-interaction monitoring for data-driven software improvement based on feature usage. While these types of requirements explicitly are articulated by stakeholders as non-functional requirements and, thus, are part of their value appraisal of the software, they do not adhere to the technical notion of metrics underlying, for example, the ISO/IEC 25010 specification [22]. As a result, they are often not clearly specified using metrics and appropriately monitored during software development. Instead, the fulfilment of these stakeholder requirements is analysed qualitatively ex-post and often outside the software engineering context.

The necessity for a categorisation of relevant metrics for value-based software engineering can also be inferred from the number of studies elaborating on the challenges arising from the misbalance of these metrics. For instance, the requirement to collect operational usage data for A/B testing [11, 23] aims on novel feature ideation to improve the value of the developed software for the customer. While the authors stress the importance of these requirements and also their heterogeneity due to the different value expectations of stakeholders, they do not present concrete metrics that allow us to quantify and monitor the fulfilment of these requirements. As an example, the customer-focussed value consideration sketched above not only requires a technical implementation for operationalising

the respective data but also an overall product strategy to also take into account business-focussed value considerations and evaluate the fulfilment of these considerations suitably. Also, maintainability requirements towards novel software features might be higher than of features having less strategic importance for a company. In summary, these works stress that in value-driven software engineering often contradictory requirements from stakeholders of different domains need to be fulfilled. Balancing these requirements presupposes defining concrete assessable fulfilment criteria based on value-oriented metrics and categories. Categorising metrics by their spanning theme could also help in balancing these contradictions in a more structured manner.

To address this challenge and to identify the metrics relevant for the stakeholders' value appraisal of software, we conducted a systematic mapping study about NFRs in DevOps [12]. Thereby, we carved out recurrent focus areas and themes of NFRs in DevOps that reflect relevant value considerations of stakeholders and derived quality metrics thereof. One important finding of this study was that only a few studies present metrics whose relevance in practice is also empirically validated. The monitoring of stakeholder objectives outside the technical domain, addressing for instance market penetration, customer satisfaction or regulatory compliance, is still hindered by the absence of a common collection of metrics for these objectives. Despite the large number of metrics mentioned in the examined papers, only a few of these metrics are empirically validated regarding their practical relevance to reflect the business, strategic, organisational or customer value of software features. Thus, the question of practical relevance of these metrics can, however, only meaningfully be answered based on an empirical investigation.

3 | RESEARCH METHODOLOGY

The foundation for this survey was the value considerations identified through the systematic mapping study. As already pointed out, one important finding of this mapping study was the apparent lack of empirical studies on the practical relevance of the value considerations described in the studies as well as appropriate indicators for evaluating the fulfilment of these value considerations. In this context, value considerations reflect important aspects of the stakeholders' value appraisal while value criteria describe concrete indicators relevant for stakeholders to judge whether these value considerations are fulfilled. As a starting point for the survey, we defined a compact set of metrics for these value criteria that can be assessed by practitioners and also used to identify potential measurement sources. Thereby, we included studies returned from the mapping study, which elaborate on the practical applicability of the herein described value criteria and addressed stakeholder objectives. Additionally, we conducted a narrative literature review to find further relevant studies giving indications for concrete value criteria. Afterwards, we grouped the resulting set of metrics into four main categories and iteratively refined them into 10 categories. For the sake of the compactness of this paper, we provide the literature references for these metrics as an online resource (https://bit.ly/3jXf2Uf).

The aim of this study was to assess the relevance of these metrics particularly from the perspective of practitioners such as software engineers, product and project managers, software testers and operational staff, as they pursue different value considerations in software engineering. Therefore, we developed an anonymous online survey targeting these stakeholders, which could be completed within about 20–25 min and distributed via professional social networks and personal invitations.

3.1 | Questionnaire design

The questionnaire for the survey consisted of 72 questions, grouped into five question blocks, and an end page and was developed based on the guidelines of Kitchenham [24]. The details of these question blocks are as follows:

- **Professional Information:** In the first block, the participants were asked about their current affiliation, the type of software they are developing/operating, and the industry sector of their division/company.
- **Roles and Experience:** In this block, the participants had to select three roles that they currently have or previously had in software engineering projects (potentially also in different companies). At least one role had to be given by each participant. For each role, the participants were asked to also give their years of experience.
- **Software Development Background:** The third block gathered information pertaining to the frequency of particular software engineering steps and the typical length of development iterations in the companies. Knowing about the frequencies of these steps is important for operationalising the examined metrics in practice. Each metric has a particular point in time or step in the development lifecycle, where it can be measured.
- **Metric Assessment:** To not overburden the participants and, in the worst case, cause them to quit the survey or make unreflected assessments, the survey showed each metric category on an individual page. The idea behind this was to keep participants motivated, as they could decide after each block whether they wanted to answer another one. Furthermore, this should ensure that each participant assesses at least one block of metrics. The sequence of metric blocks was randomly chosen by the survey for each participant. In each metric assessment block, the participants assessed the relevance of five to eight metrics on a five-point numerical scale from one (low relevance) to five (high relevance). At the beginning of each question block, the survey showed an individual description (in two or three sentences) of the corresponding metric category to the participants. When describing the rationale of a metric category, we intentionally used basic terminology to ensure a common understanding of the category for all participants regardless of their role in the software

development process. For each metric, the participants could also state that they do not know the metric and, thus, are unable to rate it. The answers to the metric assessment blocks were mandatory, meaning that participants either needed to rate all metrics or state that they do not know an individual metric.

Afterwards, the participants were asked if the presented metrics completely cover the value considerations and, thus, the criteria underlying these considerations. The answer to this question was mandatory. In case of the answer *incomplete*, the participant was asked to describe missing metrics in a free text field. Eventually, the participants were asked to textually describe possible data sources for acquiring measurements for these metrics. Answering this question could optionally be skipped.

At the end of each assessment block, the participants were asked whether they wanted to continue with another block or quit the survey. At that point, the survey showed information to the participants to explicitly express appreciation for the already spent effort and emphasised the importance for continuing assessing another set of metrics. By giving participants the freedom to leave the survey after each assessment block, the participants should be kept motivated and prevented from unthinkingly rating the metrics just to quickly finish the survey.

- **End of Survey:** On the final page, the survey again showed appreciation for the participant's time and asked whether the participant was interested in getting the final results of the survey by email. To not relate this email address with a particular participant, a link for sending an email with a given subject was displayed.

3.2 | Participant recruitment

Two approaches were followed for recruiting participants for the survey. First, by personally inviting professionals who are knowledgeable in the examined topics and, second, by promoting the survey in expert groups of professional networks such as XING (http://www.xing.com), ResearchGate (http://www.researchgate.com), and LinkedIn (https://www.linkedin.com). When personally inviting people, we always outlined the general objective of the survey and provided an access token to the survey. The access token allowed to monitor which groups of participants (associated with this token) already had answered the survey and also prevented unsolicited ratings from external people. In the personal invitation email, we also asked to distribute the link to the survey among colleagues to additionally increase participation in the survey.

For recruiting participants on the mentioned professional networks, we approached only people in expert groups that deal with the examined topics. Therefore, we described the general objective of the survey and provided the link to it as well as the access token to monitor participation and, in case of too few respondents, repeated my call for participation. In this manner, we approached 13 topic-related expert groups on LinkedIn, one in ResearchGate, three on LinkedIn, and sent personal invitations to seven groups of professionals that we personally know.

4 | DEMOGRAPHIC ASPECTS OF PARTICIPANTS

The survey was provided in the time frame between June until October 2020. In total, 81 people started the survey either by following the personal invitation link or by clicking the link in any of the expert groups in the professional social networks. Afterwards, 31 participants left the survey right before answering the first question block, resulting in 60 participants who answered the professional information block pertaining to their affiliations. Thereafter, nine participants quit the survey and the remaining 51 participants proceeded to the page asking them about their roles and experiences in their professional lives.

After answering about their roles and experiences, four participants dropped out before being asked about the software development background in their companies, and six further participants quit the survey after this question block. The remaining 41 participants successfully finished the survey, meaning that they assessed at least one block of metrics. Hence, the percentages given in this presentation relate to these 41 participants as they not only completed at least one metric assessment block but also provided enough information for an appropriate analysis in the previous question blocks. In the median, the participants answered four question blocks.

4.1 | Professional information

In the first question block of the survey, the participants gave their current affiliation and the industry sector of their companies. The majority of participants (49%) works in large companies with more than 1000 employees, followed by 24% from companies with less than 1000 employees, 14% from companies with less than 250 employees, 6% from companies with less than 100 employees, and 4% from companies with less than 50 employees. Summing up all participants from companies with 50 to 1000 employees, this group accounts for 96% of the participants. Only two percent of participants originate from academic organisations or are self-employed. As the survey explicitly aimed at gathering assessments regarding the relevance of the metrics from practitioners, this distribution also supports this objective. From the perspective of the companies' industry sectors, the vast majority of participants (49%) works in companies in the field of professional IT and software services, 14% in companies that develop or operate software for production and manufacturing, 10% for public administration, 8% for commerce, 6% for telecommunications, 6% for transportation/logistics, and equally 2% in health and education services. Four percent of participants work in other industry sectors. The survey did not allow participants to textually describe alternative industry sectors. As a result, no statement can be given about the four percent of participants having chosen *other* as their companies' industry sector.

4.2 | Background and experience

The second question block examined the background of the participants, meaning their roles and experience in these roles, potentially also in previous affiliations. Figure 1 shows the distribution of these roles in percent underneath the corresponding bar, whereby the largest shares can be attributed to software engineering roles such as software engineers (20%) and architects (17%), totalling both to 37%. Summing up the less technical roles such as consultants, product owners, project, and product managers totals to 21%. Roles having a focus on the operation of the software can be seen by infrastructure engineers, operations managers, and site reliability engineers. These roles are held by 22% of participants. Although the participants of the survey have some relation to business functions and operations, we did not achieve full coverage of all roles necessary for value orientation in software development. Therefore, the findings of this survey reflect the view of more technical participants onto value orientation but lacks especially the business-oriented side.

For each role, the participants also had to give the number of years in experience in these roles (cf. Figure 1). Interestingly, the participants have the most experience as consultants and operations managers. Thereby, it has to be noted that in the field of IT consultancy, companies often refer to software engineers as consultants. However, the survey did not further examine the duties associated with these job roles and possible similarities in the companies. In summary, the distribution shows that the participants on average have more years of experience in technical domains than in fields such as project or product management.

Also, the participants were asked which type of software product they are developing or operating. The analysis of the participants' answers to this question highlights that the majority of 45% of participants' works in the context of business information systems, 18% work on development tools, 16% on embedded systems, 10% on knowledge-based expert systems, 6% on mobile systems, and 2% on service-oriented systems. Four percent work on software outside the given fields, for example, mainframe systems. It can be depicted from these numbers that development tools and embedded systems are the two second important types of software that the participants work with.

5 | SOFTWARE DEVELOPMENT PRACTICE IN COMPANIES

The third question block examined particular aspects of the software development practice in the companies, such as the frequency of particular engineering steps during software development. When answering these questions, participants could also indicate not knowing the frequency of individual steps.

Figure 2 shows the frequency of the different steps, as known to the participants. In summary, most of the examined steps are executed on a frequent basis mostly hourly or daily in the companies. Interestingly, steps such as evaluating software security or quality reports, or performing static code analysis are never conducted in around 19% of the companies. Also when it comes to testing, around 14% of companies never conduct concrete testing steps in their development lifecycle. As expected, steps targeting developing and deploying software in many companies are executed in shorter intervals than steps targeting on evaluating software architecture, quality or security reports.

Asked about the length of the development iterations in their companies, the participants mentioned weeks (87%) and months (11%) as typical durations. Two percent of participants did not know the duration of iterations in their companies.

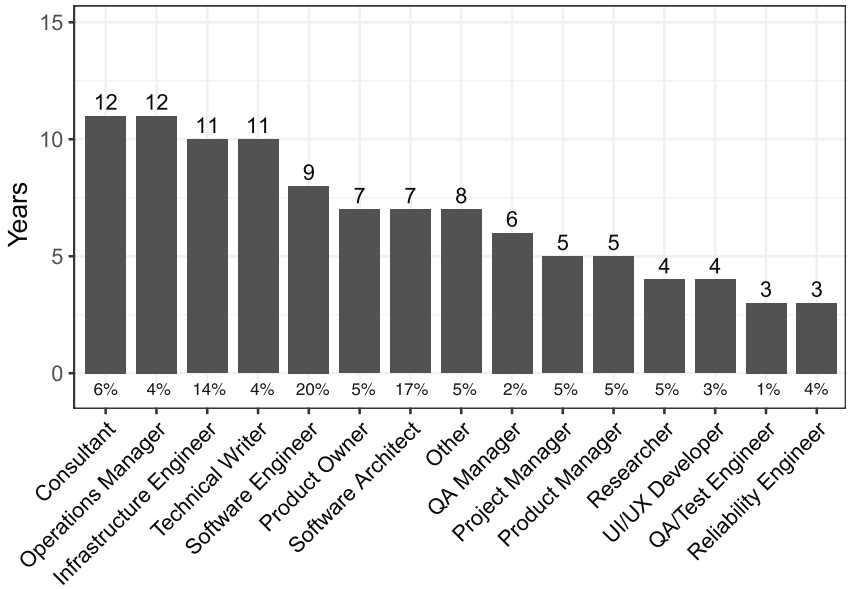

FIGURE 1 Average experience and distribution of job roles

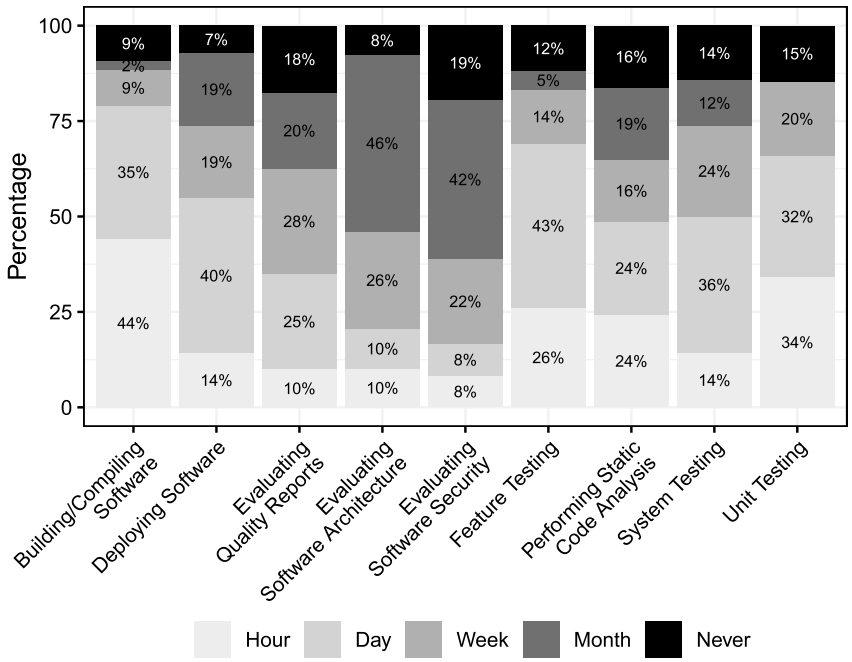

FIGURE 2 Software development steps in the companies

6 | RELEVANCE ASSESSMENT OF METRICS

Based on the assessments from 41 participants we calculated the average assessment for each of the 61 metrics. The average assessment was calculated from the individual relevance assessments ranging from 1 point (very low) to 5 points (very high) of the participants' responses. For getting a quick overview of the most relevant metrics, we ordered them by average assessment and standard deviation in decreasing order. Thereby, we regarded the practical relevance of metrics having an average assessment above 4.5 points as *very high*, an average assessment between (including) 3.5 and 4.5 points as *high*, between (including) 2.5 and 3.5 points as *moderate*, between (including) 1.5 and 2.5 as *low*, and below 1.5 as *very low*. Accordingly, the participants considered 23 metrics as highly relevant, 37 metrics as moderately relevant and one metric as little relevant in practice.

In addition, we calculated the standard deviation and skew of the individual assessments and depict the details for each metric in Table 1. The standard deviations of all metrics range between 0.94 and 1.61. A low standard deviation among assessment values can be interpreted as consensus among the participants regarding the relevance of metrics, whereas contrarily high values can be interpreted as disagreement.

The skewness relates to the symmetry of the assessments of a metric around the mean. It expresses the tendency of the metric towards being assessed as either little or highly relevant. In general, negative skewness indicates that the mean of the assessments is less than the median. Likewise, positive values indicate that the mean of the respective assessments is larger than the median. To support the interpretation, we multiplied the values of the skew with minus one so that positive values express a tendency towards high relevance and contrarily, negative values towards little relevance.

From a broader perspective, it can be observed that for metrics whose relevance is rated higher on average, also the standard deviation is smaller compared to metrics, which are rated less relevant on average. This observation suggests that the participants more likely agreed consensually upon which metrics are highly relevant in practice compared to which metrics are not relevant. Overall, the skew of the metrics indicates a tendency towards higher relevance and, thereby, especially for metrics, which already have a high average rating. Conversely, only 15 of the 61 metrics have a negative skew indicating a tendency towards low importance, whereby the corresponding metrics have been rated as moderately relevant in practice.

7 | CATEGORISATION OF METRICS

As the starting point for the categorisation of the metrics, we identified the most important aspects of business models such as *customer segments* and *customer relationships* [13, 25], *financials* [25, 26] and *value creation* and *value capture* [27–29] that shall be covered by the metrics. Based on these aspects, we grouped the metrics examined in the survey according to four main perspectives: *financial and market performance*, *customer satisfaction*, *value creation*, and *value proposition*. The metrics were identified through our preceding systematic mapping study examining NFRs, as indicators of the stakeholders' value considerations [12].

The presented categorisation of metrics along with their individual practical relevance shall support practitioners in a twofold manner: (i) in selecting purposeful metrics for the value considerations of their stakeholders in software development, and (ii) by sketching possible measurement sources for the metrics in their software projects. For researchers, the

TABLE 1 Relevance ranking of metrics

Metric	Average	Relevance	Standard Deviation/Skew (*-1)
Availability	4.16	High	1.40 / 0.97
Downtime	4.11	High	1.33 / 0.89
User count	4.00	High	1.26 / 0.93
Response time	3.94	High	1.41 / 0.65
Build duration	3.93	High	1.36 / 0.71
CPU usage	3.88	High	1.20 / 0.50
Feature test coverage	3.88	High	1.29 / 0.58
Unit test coverage	3.88	High	1.38 / 0.49
# Missed quality gates	3.83	High	1.31 / 0.71
Automated system tests ratio	3.80	High	1.26 / 0.84
# Reliability incidents	3.78	High	1.23 / 0.22
# Vulnerabilities	3.72	High	1.19 / 0.91
Complexity of architectural constraints	3.71	High	1.10 / 0.76
Complexity of features	3.71	High	1.16 / 0.81
Memory consumption	3.71	High	1.21 / 0.65
# Deployments to QA systems	3.62	High	1.39 / 0.15
Security rating	3.61	High	1.20 / 0.44
# Completed change requests	3.58	High	1.20 / 0.29
Technical debt	3.58	High	1.22 / 0.44
Customer satisfaction index	3.53	High	1.41 / 0.49
System test duration	3.50	High	1.09 / 0.66
Capacity	3.50	High	1.29 / 0.15
Task completion time	3.50	High	1.41 / 0.46
Mean time between failures	3.47	Moderate	1.12 / 0.27
# Post-release defects	3.47	Moderate	1.17 / 0.72
# Rollbacks due to technical problems	3.46	Moderate	1.20 / 0.59
# Deployments to production systems	3.46	Moderate	1.20 / 0.32
Deployment duration	3.43	Moderate	1.09 / 0.49
Unit test duration during build	3.36	Moderate	1.01 / 0.68
# Rule violations	3.33	Moderate	1.24 / 0.25
# Open change requests	3.26	Moderate	1.02 / 0.36
Development velocity	3.25	Moderate	1.13 / 0.20
Percentage of deployment failures	3.23	Moderate	1.24 / 0.16
Network usage	3.12	Moderate	1.17 / 0.43
# Usability incidents	3.12	Moderate	1.31 / 0.38
# Completed user stories	3.07	Moderate	1.16 / 0.12
Product revenue	3.07	Moderate	1.58 / 0.00
# System tests	3.06	Moderate	1.24 / 0.11
# Unit tests	3.06	Moderate	1.24 / 0.11
Degree of framework compliance	3.06	Moderate	1.39 / 0.10

(Continues)

TABLE 1 (Continued)

Metric	Average	Relevance	Standard Deviation/Skew (*-1)
Product rating in online store	3.00	Moderate	1.57 / 0.11
# Delivered stories	2.94	Moderate	0.94 / −0.10
Retention rate	2.94	Moderate	1.48 / 0.13
# Feature invocations per day	2.94	Moderate	1.48 / −0.10
Usability satisfaction index	2.93	Moderate	1.22 / −0.12
# Feature invocations (overall)	2.93	Moderate	1.49 / 0.14
Market share per customer segment	2.93	Moderate	1.56 / −0.09
Benchmarked quality	2.89	Moderate	1.23 / −0.02
Disk usage	2.88	Moderate	0.99 / 0.22
# Delivered features	2.83	Moderate	1.19 / 0.27
# Inappropriate feature rollbacks	2.82	Moderate	1.29 / −0.14
Bounce rate	2.80	Moderate	1.61 / −0.11
Churn rate	2.71	Moderate	1.54 / −0.33
# Customer segments	2.69	Moderate	1.36 / 0.16
Presence time	2.67	Moderate	1.23 / −0.05
Product revenue/ customer segment	2.64	Moderate	1.28 / −0.22
Product market share	2.57	Moderate	1.28 / −0.16
Safety regulation conformance	2.56	Moderate	1.51 / −0.29
# Planned stories	2.53	Moderate	1.29 / −0.02
# Feature invocations/customer segm.	2.50	Moderate	1.21 / −0.53
# Clicks	2.33	low	1.18 / −0.62

categorisation and the metrics' relevance assessment shall provide important information for developing measurement and assessment methodologies or tools for the practically most relevant value considerations in software development.

In the following, we depict the rating responses in percentages for each possible assessment answer on the scale from very low (1 point) to very high (5 points). Thereby, we highlight the average relevance category for each metric in black. As participants could also indicate that they do not know a specific metric, we show the percentages of participants having actually rated the corresponding metric and, thus, having answered this question, in the right block of each figure.

7.1 | Financial and market performance metrics

Financial and market performance metrics express the expected revenues and market penetration of the provided software features. Basically, this category of metrics covers indicators of the development of organisation's actual and expected monetary success with the software and can be measured in the *operation* phase in software development. Figure 3 shows the five metrics of this category ordered by decreasing relevance. In general, the participants rated each metric and on average between high and moderate (4.00–2.57 points). Also, the metrics of this category are well known to the participants, indicated by a percentage of rating answers above 82% each.

- **User count:** Number of users interacting with software features or signing in for a subscription. The overall user count is also reflected through the number of regularly billed users.
- **Product revenue:** Revenue achieved with a concrete software feature provided to customers.
- **Market share per customer segment:** Percentage of a specific customer segment on the overall market. Customers can be segmented into homogeneous groups by similar characteristics.
- **Product revenue per customer segment:** Revenue achieved with software features for a particular customer segment.
- **Product market share:** Percentage of the overall market on comparable products attributed to a specific software feature.

As already explained earlier, the answer to the question of potential data sources for the metrics was optional. Participants

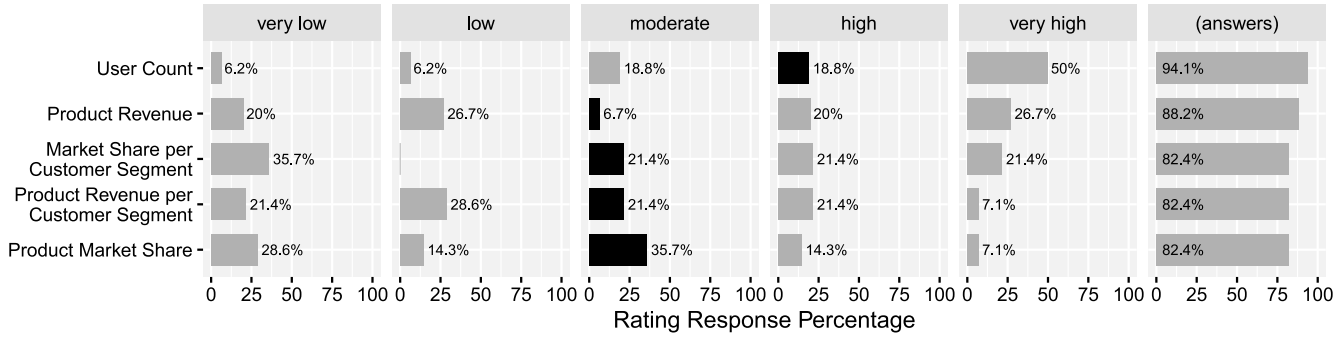

FIGURE 3 Ratings of financial and market performance metrics

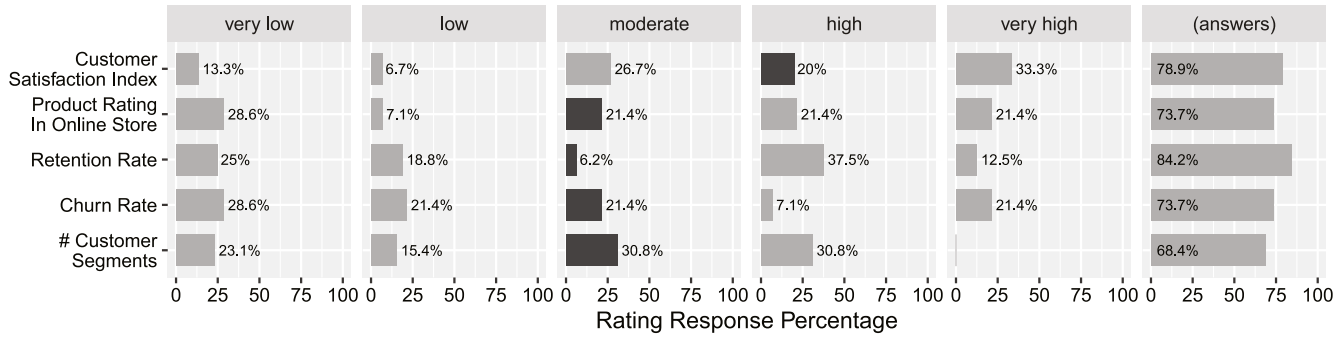

FIGURE 4 Ratings of customer satisfaction metrics

elaborating on potential sources for measuring these metrics mentioned ERP systems (and thereby *SAP* specifically), in-app purchase monitoring via dedicated APIs such as *Google Firebase* (https://firebase.google.com), and purchase monitoring through ordering systems.

7.2 | Customer satisfaction metrics

Five metrics capture the customers' satisfaction with the provided software features, the customers' segmentation, retention and churn (cf. Figure 4). The objective behind these metrics is to assess the behaviour of the customers particularly upon changes of cost, functionality, or technical characteristics of the provided software features. With the exception of the customer satisfaction index (3.53 points), which was assessed as highly relevant, the participants rated these metrics moderately relevant on average. Compared to the other categories, most metrics of this category are less known to the participants, resulting in fewer rating answers.

- **Customer satisfaction index:** The customer satisfaction index is a standardised test that examines the customers' perceived quality, expectations, value, overall satisfaction, complaints, and loyalty on a five-point scale.
- **Product rating in an online store:** In the frame of this metric, a product is understood as a bundle of different software features. Ratings typically are expressed on a numbered scale and are an important and reliable source for measuring user satisfaction. When measuring user satisfaction through online ratings in app stores, this also diminishes the problem of confirmation bias compared to, for example, personal interviews with customers.
- **Retention rate:** Customer retention rate refers to the ability of a company to retain its customers over time.
- **Churn rate:** This rate reflects how many previous customers have stopped paying for a software feature in the given period.
- **Number of customer segments:** Number of customer segments served by software features. Customer segments divide the customer base by shared characteristics of the customers, for example, objectives of the users, anticipated user experience or spending behaviour.

The participants mentioned CRM systems, customer surveys, and dedicated in-app monitoring tools as potential sources for measuring these metrics.

7.3 | Value creation metrics

Value creation metrics focus on the efficiency that software features are created and the value of these features provided to users. For instance, they address development, test, deployment, architectural, and maintainability requirements throughout the development process. They provide metrics to assess the process quality involved in fulfilling the value criteria from different stakeholders in the software product lifecycle.

This category can be refined into four subcategories, which are described below.

7.3.1 | Continuous integration and deployment metrics

This subcategory covers eight metrics, as shown in Figure 5. Based on the assessments gathered through the survey, the average relevance of these metrics ranges between high and moderate (3.93–3.23 points). Overall, the metrics are well known to participants, indicated by over 81% rating answers to each metric.

- **Build duration:** Gross build duration for software features including all necessary test steps.
- **Number of deployments to quality assurance (QA) systems:** This metric indicates how frequently code is deployed to dedicated systems, for example, integration, acceptance, or performance testing as well as to systems that mimic production systems.
- **System test duration:** Time needed for system testing as a time span within the overall build time.
- **Number of rollbacks due to technical problems:** Technical problems comprise faulty functionality, violated non-functional requirements, and other exceptional runtime behaviour of software features.
- **Number of deployments to production systems:** The metric indicates how frequently code is deployed and reaches customers.
- **Deployment duration:** Time needed to deploy software features into production.
- **Unit test duration:** Time needed for unit testing as a time span within the overall build time.
- **Deployment failures:** Percentage of failed deployments to production, staging, or quality assurance systems.

In the participants' companies, the data for the above metrics can be acquired from development environments, build servers, and issue trackers. While time-based metrics also can be acquired automatically, the number of deployments and rollbacks often needs manual counting or additional efforts. Also, many companies use commercial software tools for continuous integration and deployment, which calculates these metrics automatically.

7.3.2 | Test and development efficiency metrics

Figure 6 shows the distribution of the relevance assessments for each metric in a descending order of their average rating. The average relevance assessment ranges from high to moderate (3.88–2.53 points). Similar to the last category of metrics, these metrics are well known to participants, indicated by over 83% rating answers to each metric.

- **Feature test coverage:** Code coverage testing an individual feature.
- **Unit test coverage:** Code covered by unit tests.
- **Automated system tests ratio:** Ratio of automated to manual tests.
- **Development velocity:** Development team's speed for developing software features. Velocity is a calculation of the number of story points completed during a development iteration.
- **Number of completed stories:** per development iteration.
- **Number of system tests**
- **Number of unit tests**
- **Number of planned stories:** per development iteration.

In the survey, the participants mentioned tools of the build pipeline and dedicated integration servers as the most important streams of data for these metrics. In addition, agile

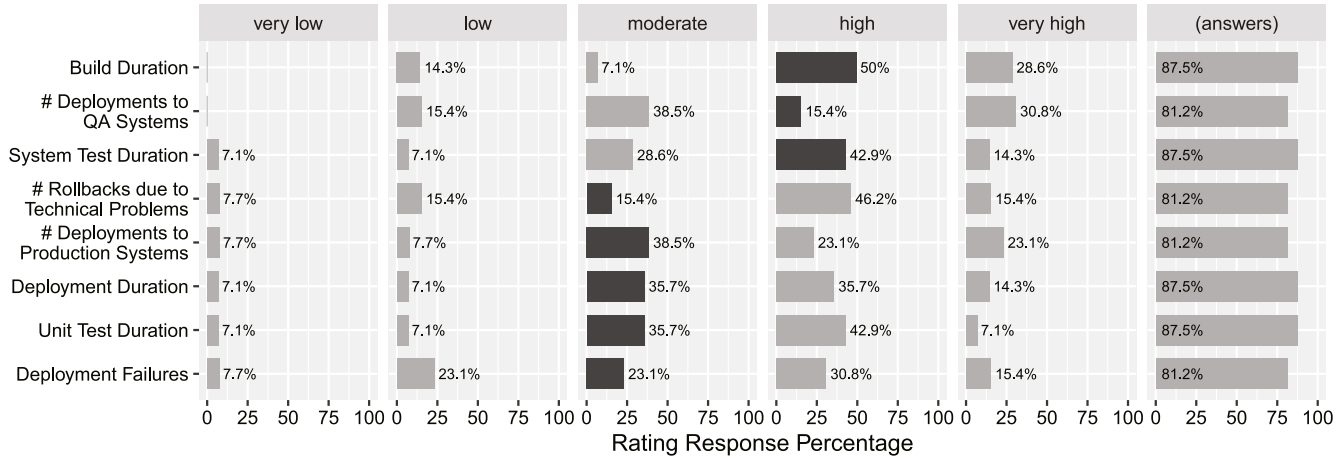

FIGURE 5 Ratings of continuous integration and deployment metrics

planning and static code analysis tools are given as a further source, particularly pertaining to monitoring development efforts and for measuring test coverage of features.

7.3.3 | Resource efficiency and architecture metrics

Seven metrics address value criteria associated with resource efficiency and architectural characteristics of the developed or operated software. Figure 7 presents the details of the relevance assessment through the participants for each metric in a descending order, whereby the average relevance ranges between high and moderate (3.88–2.88 points). Similar to the last category of metrics, these metrics are well known to participants, indicated by over 84% rating answers to each metric.

- **CPU usage:** CPU usage of virtual containers and servers operating software features.
- **Memory consumption:** Memory consumption of virtual containers and servers operating software features.
- **Complexity of features:** Number of story or function points reflecting the complexity of software features. The complexity of features must take into account functional as well as non-functional requirements.
- **Complexity of architectural constraints:** Number of architectural constraints, which, for example, arise through specific architectural styles or model-driven approaches.
- **Network usage:** Network traffic on different levels, for example, container, server, or application level.
- **Degree of framework compliance:** The degree of framework compliance can be measured, for example, from the number of used deprecated APIs or used outdated versions of third-party libraries. Third-party libraries and frameworks significantly impact the maintainability of software features.
- **Disk usage:** Disk usage on different levels, for example, virtualized container, server, or feature level.

Most participants mentioned infrastructure monitoring tools (e.g. Nagios [https://www.nagios.org/]), commercial Application Performance Monitoring (APM) systems, code analysis tools (e.g. SonarQube [https://www.sonarqube.org/])

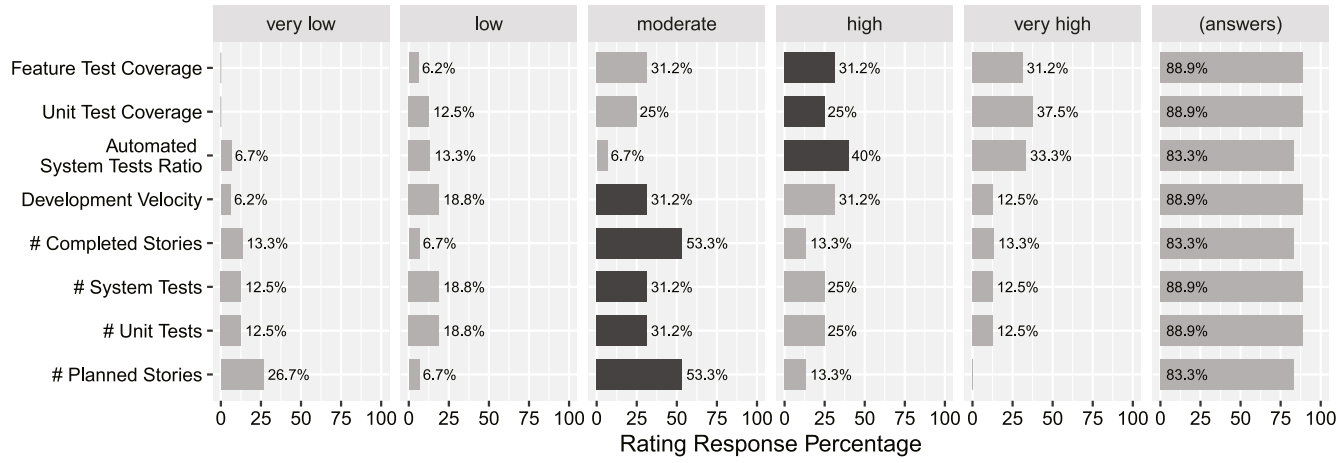

FIGURE 6 Ratings of test and development efficiency metrics

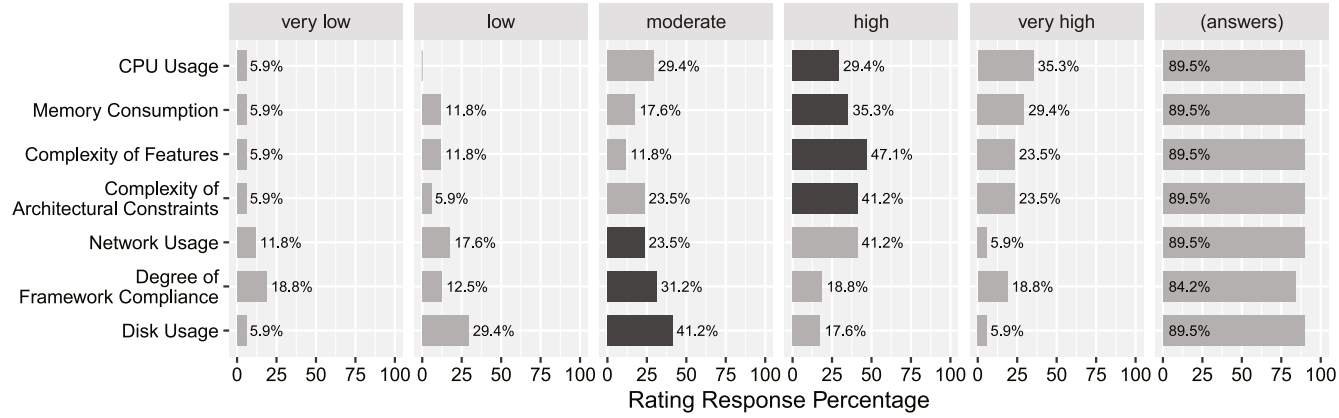

FIGURE 7 Ratings of resource efficiency and architecture metrics

as the main data source to measure these metrics. One participant mentioned that in his company log files are used in conjunction with issue trackers for a comprehensive analysis of the data underlying these metrics.

7.3.4 | Maintainability metrics

Maintainability metrics capture properties that reflect the expected effort to change, adapt, extend or correct the source code of software features. In total, we selected six metrics for this subcategory (cf. Figure 8), whereby the average relevance rating ranges from high to moderate (3.83–2.89 points). Overall, the metrics of this category are less known to the participants than the former categories, indicated by over 78% rating answers to each metric.

- **Number of missed quality gates:** Number of unfulfilled quality gates of software features.
- **Number of completed change requests:** Number of completed change requests. Change requests are regarded as completed upon successful deployment to a production system.
- **Technical debt:** Technical debt, measured for instance through the SQALE index, of software features.
- **Number of rule violations:** Number of rule violations, for example, to prevent code smells or security weaknesses or enforce coding standards.
- **Number of open change requests:** Number of open change requests. Change requests are regarded as open until successful deployment to a production system.
- **Benchmarked quality:** Quality index calculated from source code measurements of comparable projects, for example, to compare the maintainability of a project through the number of coding rule violations. The benchmarked projects must be similar in size to ensure the expressiveness of the calculated quality index.

The relevant data for these metrics can be acquired from two main sources: First, from issue trackers and project management tools, and second, from code analysis, dedicated commercial code refactoring tools, and development environments in general. One participant mentioned that change requests are derived from open GitHub issues. Another participant explicitly stressed that they manually assess technical debt in the respective company.

7.4 | Value proposition metrics

Value proposition metrics focus on the benefits and additional value offered to users through software features. For instance, they express technical characteristics such as performance, reliability, or security but also the usability of software features and their suitability to solve the users' problems. In total, they address relevant criteria of the users' value appraisal of software features.

7.4.1 | Security and performance metrics

Five metrics capture security and performance characteristics of software features, as illustrated in Figure 9. The average relevance of these metrics ranges between high (near the boundary to very high) and moderate (3.94–2.56 points). Apart from the metric related to the conformance to safety regulations the other metrics are well known to the participants and received over 85% of rating answers each.

- **Response time:** Response time of software features to serve inbound requests in a particular time frame.
- **Number of vulnerabilities:** Number of security vulnerabilities in the source code of software features.
- **Security rating:** Security rating of software features and services on an ordered scale. It can be calculated from the number of vulnerabilities and their criticality.
- **Capacity:** Number of inbound requests towards software features and services that can be handled in a particular time frame. It defines the maximum rate of requests that can be served in this time frame.
- **Safety regulation conformance:** Conformance of software features and services with a specific Safety Integrity Level (SIL) of the IEC 61508 standard [30]. The selection of the SI level is product-specific and depends on the intended

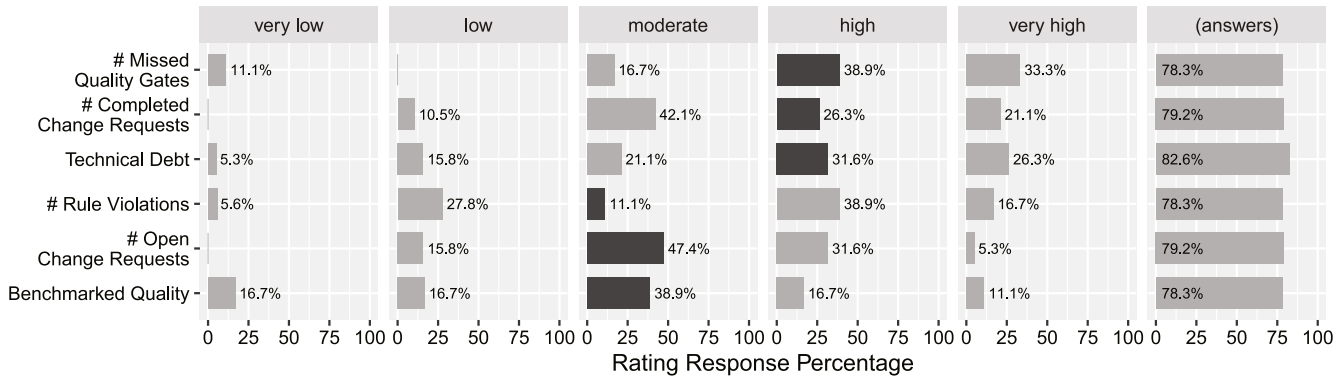

FIGURE 8 Ratings of maintainability metrics

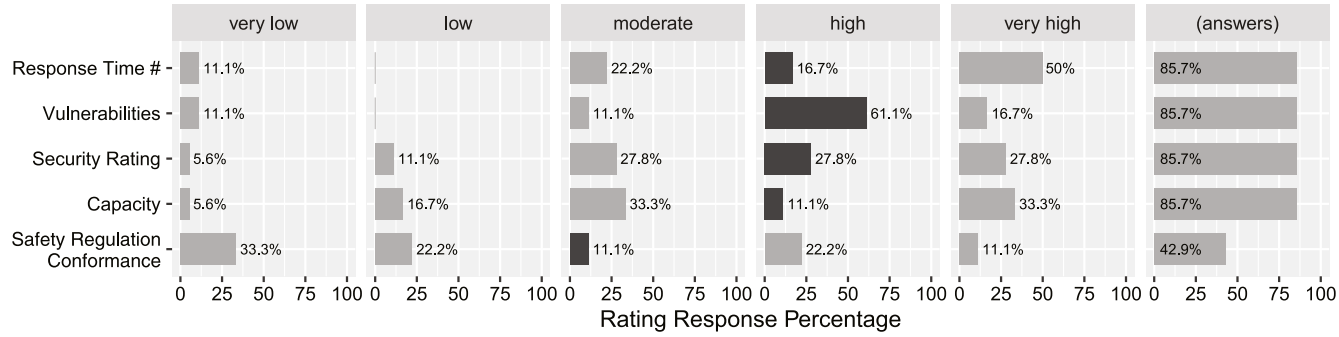

FIGURE 9 Ratings of security and performance metrics

use case. The required measures to assess the conformance with this standard must be selected individually based on the development process.

When it comes to performance metrics, there was consensus among the participants that the respective data can be acquired from APM tools in their companies. Pertaining to security-related metrics, multiple sources were mentioned, such as static code analysis tools, security scanners, results from penetration tests and manual code reviews.

7.4.2 | Feature reliability metrics

Five metrics take different perspectives onto feature reliability as one important aspect of the customers conceive of a feature's value. Figure 10 illustrates the distribution of the participants' relevance assessment of these metrics, which on average ranges between very high and moderate (4.16–3.47 points). This category of metrics is best known to the participants and each metric received over 85% of rating answers.

- **Availability:** Percentage of time that software features provide their functionality and are able to handle inbound requests. As an example, availability can demand that a software feature is able to respond 99% of the time in a month, even with change in user volume.
- **Downtime:** Downtime of software features in a particular time frame. Downtime is the inverse measure of availability.
- **Number of reliability incidents:** Number of incidents impairing the operation of software features and services.
- **Mean Time Between Failures (MTBF):** Mean time between failures of software features in a particular time frame.
- **Number of post-release defects:** Number of defects becoming apparent in the productive environment after testing. A post-release defect is a user-reported problem in a software release, which needs to be fixed.

Participants giving concrete sources for measurement of these data predominantly mentioned software monitoring tools as the prime source. In the second place, participants mentioned issue and bug trackers, followed by log files generated by the different systems as relevant sources to operationalise the metrics.

7.4.3 | Usability metrics

Usability metrics address the value appraisal through the user in the direct interaction with the software. Figure 11 shows the relevance assessments through the participants in a decreasing order, which range on overage between highly and low relevance (3.50–2.33 points). Concretely, in the survey, the number of clicks on the user interface has been rated as the metric with the lowest practical relevance. In total, the metrics of this category are known to 78% participants, which is slightly smaller than that of the previous categories. Especially, the relevance of the metric related to the (user) presence time could be answered by fewer participants.

- **Task completion time:** Time needed to complete representative and predefined tasks with software features. It measures the whole time needed to complete the task, regardless of success or failure.
- **Number of usability incidents:** Number of user reported usability incidents of software features. The numbers can be collected from post-test questionnaires with end-users or by analysing online forums.
- **Usability satisfaction index:** Usability satisfaction index, measured, for example, through the *System Usability Scale (SUS)* [31], which is a standardised test that is easily understood by a wide range of people and whose results can be analysed also through people who have little or no experience in human factors and usability. It assesses user satisfaction through 10 questions that are answered on a five-point Likert scale.
- **Bounce rate:** Bounce rate when using software features. Bounce rate is the number of users of a software offering that just use the entry point of a software workflow without using any further features of the workflow.
- **Presence time:** Presence time per user on the software system using software features.
- **Number of clicks:** Number of clicks per user in the user interface of software features.

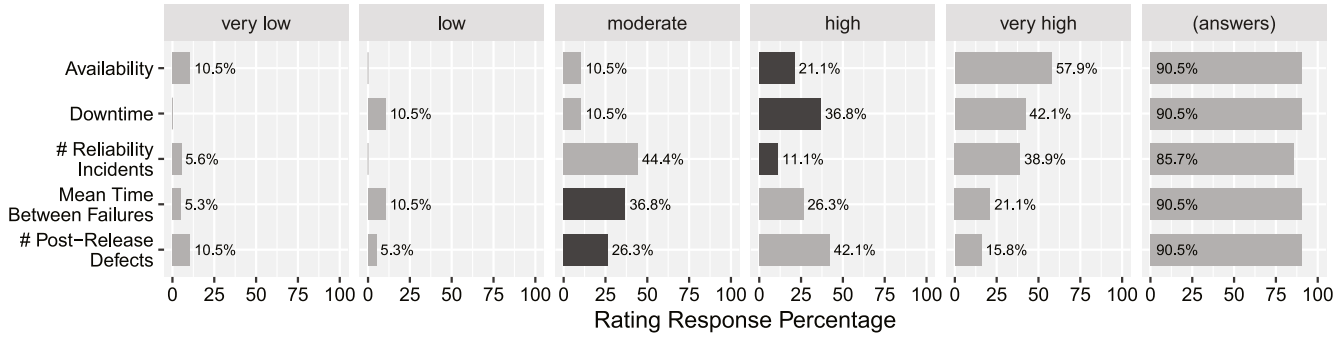

FIGURE 10 Ratings of feature reliability metrics

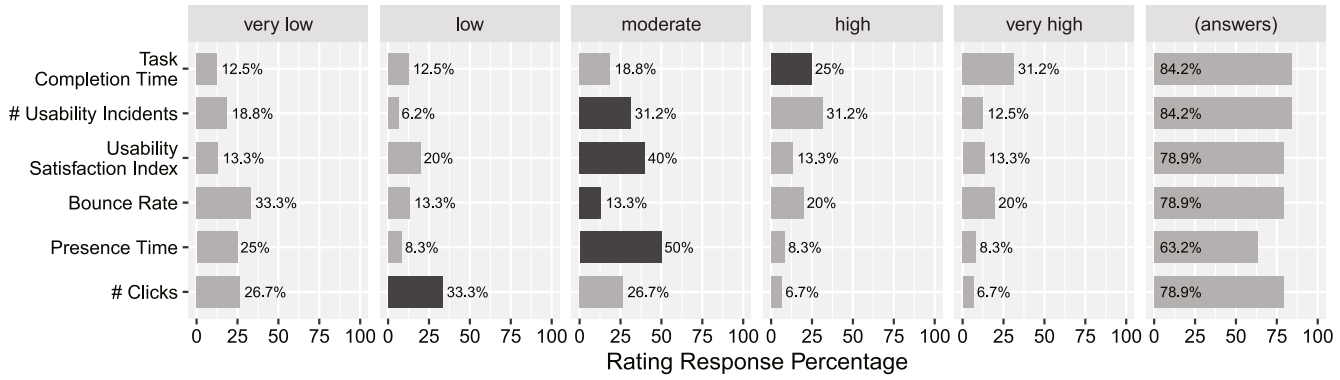

FIGURE 11 Ratings of usability metrics

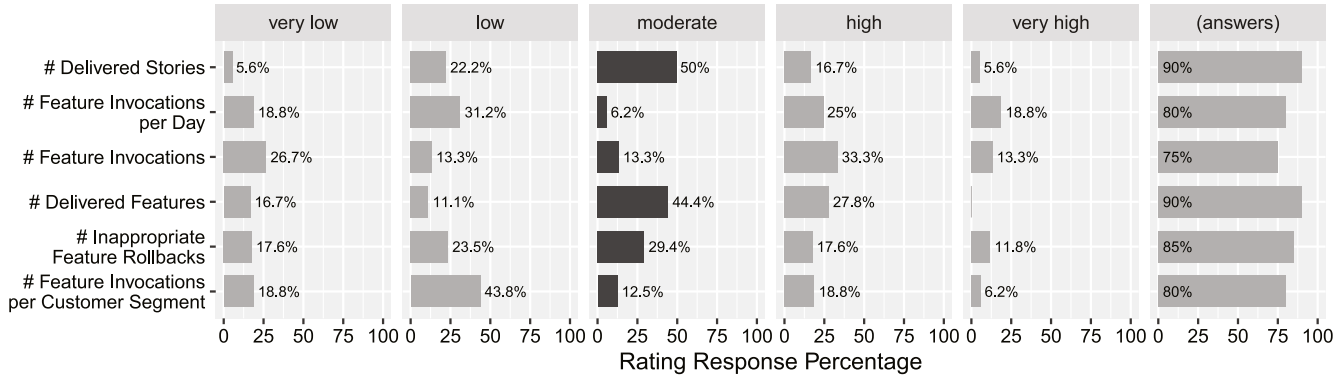

FIGURE 12 Ratings of functional appropriateness and improvement metrics

Asked about measurement sources, the participants mentioned that these metrics are not automatically measured in their companies. However, user interaction with critical parts of the software is logged to files and analysed via spreadsheets.

7.4.4 | Functional appropriateness and improvement metrics

Finally, six metrics focus on functional appropriateness and improvement of software, from the customer's perspective, as illustrated in Figure 12. For instance, they capture the number of delivered stories and features or, contrarily, the number of revocations of inappropriate features through rollbacks. The participants rated the relevance of these metrics on average as moderate (2.94–2.50 points). In general, with a percentage of rating answers above 80% each (apart from the metric related to the number of feature invocations), the metrics of this category are well known to the participants.

- **Number of delivered stories**
- **Number of feature invocations per day:** Number of invocations (from all users) of software features per single day.

This number might vary among the hours of the day and also among the days of the week.

- **Number of feature invocations (overall):** Overall number of invocations (from all users) of software features per (company/product) specific time frame.
- **Number of delivered features**
- **Number of inappropriate feature rollbacks:** Number of rollbacks of inappropriate features. Features might be inappropriate because of missed technical qualities, failed deployments and integration characteristics or because of unsatisfying A/B or usability test results.
- **Number of feature invocations per customer segment**

Dedicated user monitoring and APM tools are mentioned as potential data sources in the companies for monitoring feature invocations. For operationalising the other metrics, agile project management tools and issue trackers were mentioned as potential sources.

8 | COMPLETENESS ASSESSMENT OF METRICS

8.1 | Quantitative assessment

Finally, for each metric category, the survey asked if the list of metrics completely covered a particular category or alternatively, if important metrics were missing. In that case, the participant was prompted to describe the missing metrics textually. Table 2 shows the percentage of participants having assessed a particular category as completely covered by the associated metrics.

It can be depicted from the above list that, according to the participants, two categories of metrics are completely covered by metrics and, thus, no relevant metrics are missing. To identify further relevant metrics, we analysed the textual responses of the participants describing the missing metrics qualitatively.

TABLE 2 Completeness assessment of the metrics

Metric category	Completeness (%)
Financial and market performance	100.00
Usability	100.00
Customer satisfaction	94.74
Maintainability	91.30
Security and performance	90.48
Continuous integration and deployment	87.50
Feature reliability	85.71
Functional appropriateness and improvement	85.00
Resource efficiency and architecture	84.21
Test and development efficiency	83.33

8.2 | Qualitative analysis of responses

For the categories *financial and market metrics* and *usability metrics*, no qualitative data was analysed, as the participants already assessed these categories as complete. Below we summarise the participants' responses regarding missing metrics for the remaining metric categories.

- **Customer Satisfaction Metrics**
  - Time to market
- **Maintainability Metrics**
  - Degree of compliance with operational rules
  - Degree of the fulfilment of quality gates
  - Number of committing developers per change request (Reflect complexity and scope of efforts to realise change requests)
  - Number of code lines changed per change request
- **Security and Performance Metrics**
  - Time to fix a performance impairment
  - Page load time (time to load an app screen or a web page)
  - User action time (time to complete an interaction)
  - Response time under standard and peak load
- **Continuous Integration and Deployment Metrics**
  - False-negative/positive rate of unit or systems tests
  - Time to error detection (time between initial error occurrence and its perception by software engineers)
  - Rollback time (time needed to rollback faulty deployments)
- **Feature Reliability Metrics**
  - Meantime to repair/recover (MTTR)
  - Reliability of third-party components and external services
  - Failure rate under standard and peak load
- **Functional Appropriateness and Improvement Metrics**
  - Feature requests per product segment
  - Feature adoption rate per customer segment
  - Conversion rate
- **Resource Efficiency and Architecture Metrics**
  - Database activity
  - Number of concurrent users
  - CPU time per request
  - Garbage collector suspension time (Java only)
  - Fan-In/Fan-Out complexity of features
  - Transactions per resource utilization (for instance, this metric assesses the number of possible transactions within one hour on a server instance with 1 GB of RAM)
- **Test and Development Efficiency Metrics**
  - Number of delivered story points
  - Number of open issues
  - Mean issue resolution time
  - Number of load tests
  - Performance test coverage

Participants describing missing metrics did this is on a very detailed level. Also, it can be recognised that there was particular interest in detailed metrics in the field of resource efficiency and architecture metrics and only little when it

comes to customer satisfaction. This also goes along with the observation that participants on average rated this metric category as moderately relevant.

9 | DISCUSSION

From a practical perspective, the assessments gathered from the survey participants are an important indicator of the relevance of these metrics in software development. They shall support other practitioners in selecting relevant metrics for value considerations in the software product lifecycle and sources for monitoring these metrics in their projects. As the presented categorisation provides metrics for different stakeholders of the software value chain, it can be especially relevant when practically applying value-based principles to software development contexts. Optimally, one metric from each category shall be evaluated continuously in practice to gather a comprehensive picture about how well the developed software is meeting the stakeholders' expectations. Based on the results from the survey and the different efforts for operationalising each metric in practice, the following seven metrics can serve as an initial selection in practice: *user count*, *build duration*, *automated system test ratio*, *complexity of features*, *number of missed quality gates*, *number of security vulnerabilities*, and *downtime*.

From a research perspective, the presented metrics shall serve as a foundation for future empirical case studies, for example, focussing on the practical suitability of these metrics for value-oriented software engineering, or domain-specific surveys with larger sample sizes.

10 | THREATS TO VALIDITY

The first internal threat to the validity of this study might originate from the different understandings of the metrics through the participants. To mitigate this threat, we described each metric in a form that does not require any specific knowledge to ensure a common understanding. Particularly this pertains to metrics, which are not widely adopted. Therefore, each metric and its description were revised by two researchers to ensure a uniform understanding through the participants. We discussed and refined the metrics' descriptions based on the usual terminology found in the literature, until both researchers agreed on a common description and definition. Further, the perceived relevance of a metric might depend on the typical roles and activities of a participant. As the participants of the survey belong more to technical-oriented roles and less to business-oriented roles, the results reflect more a software engineering perspective on value orientation and not a business-oriented view.

Another confounding variable comes from the participants' understanding of the objectives behind the metric categories. To mitigate this threat, we intentionally abstained from naming concrete practices, techniques or tools in the description of the metric categories to prevent biased judgements. However, different understandings of the metric categories might have influenced the extent that participants were able to identify missing metrics for a particular category and appropriately describe them. Therefore, for identifying missing metrics, the survey provided a simple text field so that the participant could describe it in his or her words. Also, as participants could quit the survey after each question block of a category, metrics which had not been shown until that point could have been interpreted as missing. Furthermore, no comments given by the participants in the text fields indicate any form of misunderstanding affecting the participants' assessment of the metrics and of the completeness of the metric categories.

A further factor influencing the participants' relevance assessment of the metrics originates from their actual knowledge of an individual metric. To mitigate this threat, participants could answer that they do not know the metric and, thus, cannot assess its relevance. In total, only three metrics (number of customer segments, safety regulation conformance, and presence time) are known by less than 70% of the participants, which might influence the significance of the average relevance of these three metrics.

A threat to external validity can be seen in the dominance of participants coming from the industry sector of professional IT services and being affiliated with large-scale companies with over 1000 employees. This might affect the relevance of the results for other domains such for instance public administration or telecommunications. However, as many metrics require a market- and customer-oriented point of view, we regard this as minimal pertaining to the relevance of the findings for different industry sectors. Contrarily, given the affiliations of the participants primarily to large-scale companies, this could affect the applicability of the results for smaller companies and startups.

11 | CONCLUSION AND FUTURE WORK

In this work, we presented the results from an online survey with practitioners to assess the practical relevance of 61 metrics for particular value criteria and also identify possible sources for their measurement. The metrics were extracted from the papers analysed in our systematic mapping study [12], which unveiled the lack of empirical data regarding the practical relevance of value-oriented metrics. In order to define the metrics using common terminology and regard particularities important in practice, we also conducted a complementary narrative literature search. Thereby, only papers which let assume that the herein described metrics are practically usable were selected. The selected metrics were grouped into 10 categories based on four main perspectives onto value proposition, value creation, customer satisfaction and business-related criteria.

The ratings of the metrics are based on the responses of 41 participants who assessed at least one of the 10 metric categories. The participants primarily work in large-scale companies with more than 1000 employees in professional IT services as software engineers, architects, and consultants. On

average, the participants rated metrics of the categories feature reliability, feature performance, and test and development efficiency the highest. This is followed by metrics for continuous integration and deployment, resource efficiency, and maintainability. Metrics for the financial and market, usability and customer satisfaction criteria show a tendency towards being less relevant in practice. Nevertheless, this could also be a result of the technical orientation of our survey participants. For metrics assessing financial and market, and customer satisfaction criteria, the most relevant sources for measuring these metrics are ERP and CRM systems and dedicated in-app monitoring APIs. For metrics related to the software engineering process, relevant metrics can be acquired from build servers, code analysis tools, agile project planning software, issue and bug trackers, version control systems and APM systems.

Asked about the completeness of the categories' metrics, the participants rated the categories of financial and market and usability metrics as complete. For the remaining categories, it can be observed that participants missed further detailed metrics for resource efficiency, software architecture, test and development efficiency. These findings might be attributed to the fact that most of the participants have a strong technical background. An interesting finding, before the background of meeting value considerations from a business and customer perspective, is that participants missed a metric (*conversion rate*) to monitor the impact of introducing a new software feature onto business criteria such as product revenue.

Future work shall concentrate on operationalising these metrics from software projects of different industry domains and examining their suitability to monitor the different stakeholders' objectives. An interesting aspect, thereby, relates to examining differences in the perceived relevance of these metrics and linked challenges between practitioners and researchers. Therefore, future studies shall focus on equal numbers of participants of these two groups and larger numbers of participants, in general, to examine the differences between these perspectives and improve the generalisability of the results. Also, further empirical knowledge needs to be gathered from the business-oriented perspective on software products. Here, it is especially important to understand the needs of business people.

ACKNOWLEDGMENTS
None.

CONFLICT OF INTEREST
None.

PERMISSION TO REPRODUCE MATERIALS FROM OTHER SOURCES
None.

DATA AVAILABILITY STATEMENT
The data that support the findings of this study are openly available at https://bit.ly/3jXf2Uf.

ORCID
Philipp Haindl https://orcid.org/0000-0001-6075-5286

REFERENCES

1. Wohlin, C., Aurum, A.: Criteria for selecting software requirements to create product value: an industrial empirical study. In: Biffl, S., et al. (eds.) Value-based software engineering, pp. 179–200. Springer, Berlin (2006)
2. Nunamaker, J., et al.: Special issue: enhancing organizations' intellectual bandwidth: the quest for fast and effective value creation. J. Manag. Inf. Syst. 17(3), 3–8 (2000)
3. Martínez-Fernández, S., et al.: Towards automated data integration in software analytics. In: Proceedings of the International Workshop on Real-Time Business Intelligence and Analytics. BIRTE'18, pp. 6:1–6:5. ACM, New York (2018)
4. Martínez-Fernández, S., et al.: Continuously assessing and improving software quality with software analytics tools: a case study. IEEE Access. 7, 68219–68239 (2019)
5. Forsgren, N., Kersten, M.: DevOps metrics. ACM Queue. 15(6), 1–16 (2018)
6. Bonacchi, M., Perego, P.: Customer analytics: definitions, measurement and models. In: Bonacchi, M., Perego, P. (eds.) Customer accounting: creating value with customer analytics. SpringerBriefs in accounting, pp. 13–35. Springer International Publishing, Cham (2019)
7. Fabijan, A., Olsson, H.H., Bosch, J.: Early value argumentation and prediction: an iterative approach to quantifying feature value. In: Abrahamsson, P., et al. (eds.) Product-focused software process improvement, pp. 16–23. Springer International Publishing (2015)
8. Park, Y.J., et al.: Customer satisfaction index measurement and importance-performance analysis for improvement of the mobile RFID services in Korea. In: PICMET'08 - 2008 Portland International Conference on Management of Engineering Technology, pp. 2657–2665. IEEE, Cape Town (2008) ISSN: 2159-5100
9. Accenture Global Services Ltd.: Customer profitability and value analysis system. US Patent 8428997B2, 2013. Available from https://patents.google.com/patent/US8428997B2/en
10. Olsson, H.H., Bosch, J.: The HYPEX model: from opinions to data-driven software development. In: Bosch, J. (ed.) Continuous software engineering, pp. 155–164. Springer, Cham (2014)
11. Kevic, K., et al.: Characterizing experimentation in continuous deployment: a case study on Bing. In: 2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP), pp. 123–132. Buenos Aires (2017)
12. Haindl, P., Plösch, R.: Focus areas, themes, and objectives of non-functional requirements in DevOps: a systematic mapping study. In: 2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pp. 394–403. IEEE, Portorož (2020)
13. Osterwalder, A., et al.: Value proposition design: how to create products and services customers want. Strategyzer (2014)
14. Boehm, B., Huang, L.G.: Value-based software engineering: a case study. Computer. 36(3), 33–41 (2003)
15. Gruhn, V., Schäfer, C.: BizDevOps: because DevOps is not the end of the story. In: Fujita, H., Guizzi, G. (eds.) Intelligent software methodologies, tools and techniques. Communications in computer and information science, pp. 388–398. Springer International Publishing (2015)
16. Forbrig, P., Dittmar, A.: Integrating HCD into BizDevOps by using the subject-oriented approach. In: Bogdan, C., et al. (eds.) Human-centered software engineering. Lecture notes in computer science, pp. 327–334. Springer International Publishing (2019)
17. Urbach, N., et al.: The impact of digitalization on the IT department. Bus. Inf. Syst. Eng. 61(1), 123–131 (2019)
18. RiunguKalliosaari, L., et al.: DevOps adoption benefits and challenges in practice: a case study. In: Abrahamsson, P., et al. (eds.) Product-focused software process improvement. Lecture notes in computer science, pp. 590–597. Springer International Publishing (2016)
19. Senapathi, M., Buchan, J., Osman, H.: DevOps capabilities, practices, and challenges: insights from a case study. In: Proceedings of the 22nd

International Conference on Evaluation and Assessment in Software Engineering 2018. EASE'18, pp. 57–67. ACM, New York (2018)
20. Di.Nitto, E., et al.: A software architecture framework for quality-aware DevOps. In: Proceedings of the 2nd International Workshop on Quality-Aware DevOps, pp. 12–17. ACM, Saarbrücken (2016)
21. Shahin, M., et al.: Beyond continuous delivery: an empirical investigation of continuous deployment challenges. In: Proceedings of the 11th ACM/IEEE international symposium on empirical software engineering and measurement, pp. 111–120. IEEE Press (2017)
22. ISO/IEC 25010: Systems and software engineering – systems and software Quality Requirements and Evaluation (SQuaRE) – system and software quality models. International Organization for Standardization, Geneva (2011)
23. Olsson, H.H., Bosch, J.: From requirements to continuous re-prioritization of hypotheses. In: 2016 IEEE/ACM International Workshop on Continuous Software Evolution and Delivery (CSED), pp. 63–69. ACM, Austin (2016)
24. Kitchenham, B., Pfleeger, S.L.: Personal opinion surveys. In: Shull, F., Singer, J., Sjøberg, D.I.K. (eds.) Guide to advanced empirical software engineering, pp. 63–92. Springer (2008)
25. Osterwalder, A., Pigneur, Y.: Business model generation: a handbook for visionaries, game changers, and challengers, 1st ed. John Wiley and Sons, Hoboken, NJ (2010)
26. Tikkanen, H., et al.: Managerial cognition, action and the business model of the firm. Manag. Decis. 43, 789–809 (2005)
27. Saxena, K.B.C., Deodhar, S.J., Ruohonen, M.: Business model innovation in software product industry: bringing business to the bazaar, 1st ed. Springer, New York (2016)
28. Jørgensen, S., Pedersen, L.: Redesign rather than standstill. In: Jørgensen, S., Pedersen, L. (eds.) RESTART sustainable business model innovation. Palgrave studies in sustainable business in association with future earth, pp. 55–74. Springer International Publishing, Cham (2018)
29. Ebert, C., Brinkkemper, S.: Software product management – an industry evaluation. J. Syst. Softw. 95, 10–18 (2014)
30. IEC 61508: Functional safety of electrical/electronic/programmable electronic safety-related systems. International Electrotechnical Commission, Geneva (2010)
31. Brooke, J.: SUS: a quick and dirty usability scale (1996)

How to cite this article: Haindl, P., Plösch, R.: Value-oriented quality metrics in software development: practical relevance from a software engineering perspective. IET Soft. 16(2), 167–184 (2022). https://doi.org/10.1049/sfw2.12051