standards

MDPI

Article

Finest Magic Cloth or a Naked Emperor? The SKQuest Data Set on Software Metrics for Improving Transparency and Quality

Christian R. Prause [1,*] and Ralf Gerlich [2]

[1] German Aerospace Center (DLR), German Space Agency, 53227 Bonn, Germany
[2] Dr. Rainer Gerlich System and Software Engineering, 88090 Immenstaad, Germany
* Correspondence: christian.prause@dlr.de

Abstract: Software development has a problem with transparency/visibility. As an intangible product, software and its intermediate development results are hard to see or touch. Customers of custom software have difficulties checking progress, and risk coming out with costly but low-quality software. In the space domain with its often expensive and one-of-a-kind devices, which are developed in complex multitier supply chains, the risk is even greater. This paper presents the SKQuest data set. It contains the completed responses with 190 variables from an empirical study with over 100 software experts. The data set covers distinct aspects of measuring metrics and transparency in software projects. To show what information lies in the data set, the paper investigates, and affirms, from different perspectives, the following questions: Is transparency a problem in software development projects? Is there a desire for more transparency in projects? Can metrics contribute to improving the situation? Moreover, it attempts to replicate the results of an earlier study. The main contribution of this paper is, however, the SKQuest data set that is published with this paper in CSV formatas. It is a tool that enables systematic investigations of software metrics and allows research on how they can improve the efficiency of the software lifecycle, not limited to, but particularly with respect to transparency. Consequently, the paper may serve as a starting point for future research avenues in academia and industry and help to improve existing and future standards in software development.

Keywords: software metrics; software measurement; key performance indicators; project management; software quality; empirical study; survey; aerospace; ECSS

Citation: Prause, C.R.; Gerlich, R. Finest Magic Cloth or a Naked Emperor? The SKQuest Data Set on Software Metrics for Improving Transparency and Quality. *Standards* **2023**, *3*, 136–168. https://doi.org/10.3390/standards3020012

Academic Editors: Antonia Stefani and Bill Vassiliadis

Received: 24 January 2023
Revised: 1 March 2023
Accepted: 14 March 2023
Published: 4 May 2023

1. Introduction

Spaceflight missions are often long-lasting projects. Not only can flight times take several decades—e.g., the Voyager 1 and 2 space probes were launched in 1977 and have only recently crossed the border to interstellar space—but also the mission design and engineering can take well over a decade. For example, the EnMap hyperspectral remote sensing mission was approved by the German space agency in 2006 after a competitive definition phase [1] and successfully launched on 1 April 2022.

The long building times of space systems are characterized by intense engineering activities. They involve diverse partners that are entangled with one another in complex supply chains. Many space systems are one-of-a-kind devices. This means that systems must be right the first time. To assure project success and product quality, space agencies closely collaborate with their prime contractors and subcontractors that build the systems. An important ingredient of this collaboration between customers and suppliers is transparency [2], or "visibility" as it is sometimes called (cf. [3]).

Transparency usually means that information is visible to those with a stake in it. In software engineering, it relates to whether a product or development process is visible to stakeholders so that they can evaluate a software system and make decisions [4]. We follow the definition of Tu et al. who define transparency as "the degree to which stakeholders

can answer their questions by using the information they obtain about a software system during its life cycle" [4].

However, transparency in software development is notorious for being difficult to achieve. In 1972, the recently deceased famous software engineering researcher Barry Boehm recorded an Airforce decision maker's statement that might still be valid today:

> *"You software guys are too much like the weavers in the story about the Emperor and his new clothes. When I go out to check on a software development the answers I get sound like, 'We're fantastically busy weaving this magic cloth. Just wait a while and it'll look terrific.' But there's nothing I can see or touch, no numbers I can relate to, no way to pick up signals that things aren't really all that great. And there are too many people I know who have come out at the end wearing a bunch of expensive rags or nothing at all".* [5]

Since software is intangible, customers may have a hard time establishing whether developers are weaving software as if it were fine (invisible) magic cloth, or if the customer will, in fact, end up naked, i.e., with no usable results from the project, just as the emperor in Hans Christian Andersen's classical folktale.

Now, 50 years later, there is an indication that the Airforce decision maker's statement is no less valid today [6]. In fact, transparency is regarded as the primary success factor of projects [7]. Donaldson and Siegel—drawing on the Titanic incident as a metaphor—see transparency (or "visibility") as the one thing that is needed to stay clear of the icebergs of the unknowns in software systems development [3]. While several methodical tools improve transparency between customer and supplier, for example, agile development [7] and reviews [3], software measurement (or metrication/metrics), too, has the goal of creating transparency; or has it?

This paper publishes the SKQuest data set obtained through a survey aimed at investigating this and related questions. It contains 114 complete (plus 174 incomplete) responses, and an additional 117 manually checked and flagged responses that are potentially invalid. Each response combines a multitude of different items and perspectives gathered from about 50 questions, resulting in up to 200 variables per response. The valid responses represent 85 h of interviewees' time and are founded on a total of almost 1500 years of participants' domain experience.

This paper contributes:

- The SKQuest data set that is provided as supplementary material to this paper,
- The survey instrument (see Section 2 and Appendix A),
- Key figures of the SKQuest data set (see Section 3.1),
- A demonstration of the data set by means of investigation using its data:
  - If transparency is a problem in software projects,
  - If there is a desire for more transparency,
  - Whether metrics can improve transparency,
  - How a metric tool fits into the existing tool landscape (Section 3.2).
- An attempt to replicate findings from a smaller study [6] regarding how the participants' role and situation influence the perception of metrics with the larger SKQuest data set (see Section 3.3),
- A discussion of the above findings and the SKQuest data set (see Section 4).

Finally, we conclude (see Section 5).

2. Materials and Methods

This section provides further background information on the SKQuest survey and the overall motivation for conducting it, describes how the survey was conducted, including the recruitment of participants, and presents an overview of the survey instrument. Detailed information on the variables in the data set is given in Appendix A.

2.1. Survey Background

The European Cooperation for Space Standardization (ECSS) publishes a comprehensive set of European space flight standards. It covers all areas of spaceflight projects with regard to management, engineering, and quality. The ECSS standards are used extensively in many European space projects.

ECSS-Q-ST-80C [8] is the software quality standard of the ECSS. It is superordinate to ECSS-Q-HB-80-04A [9], a supplementary handbook that defines and explains the software measurement process based on ISO 15939 [10]. It contains a set of over 40 software metrics that are "based on the applicable [ECSS-Q-80] requirements complemented by the practical experience of the working group members accumulated in real projects". The metrics described in the document address the process as well as product aspects. For example, the "SPR/NCR trend analysis" is a process metric that provides a representation of the evolution of problem reports. "Cyclomatic complexity" is a product metric that indicates code complexity.

The AENEAS project (cf. [6]) is an ongoing endeavor of the German space agency to establish the use of software metrics for managing projects. It seeks to provide a standardized approach to measuring, gathering, and transmitting software metrics to improve transparency in projects across the customer–supplier boundary. At the same time, the project tries to limit the additional burden put on projects resulting from the measuring of metrics.

2.2. Original Goals of the Survey and the Four Primordial Questions

The SKQuest survey is part of the AENEAS project (cf. [6]) that develops a software infrastructure to collect and archive software metrics. The AENEAS project is part of the space agency's continued efforts to improve its processes (refer to [2] for the cross-company product quality assurance approach). In the frame of the AENEAS project, the SKQuest subproject aims to create a scientific justification and basis. It investigated how metrics can improve transparency across customer–supplier borders in space projects. Originally, four questions drove the design of the SKQuest survey:

- Is transparency a problem in software projects?
- Is there a desire for more transparency in such projects?
- Can metrics contribute to improving the situation?
- How can AENEAS fit into the existing tool landscape?

While these four questions were at the heart of the survey design, the design underwent several iterations and extensions. Hence, we consider the above four questions as *primordial questions*. The resulting survey instrument contained several more questions that captured many more facets in the context of the original questions.

2.3. Conduction of the Survey

The study was performed as a questionnaire-based structured interview using the online tool LamaPoll [11]. There was only an English version of the questionnaire. While English is not the mother tongue of many participants, it is the lingua franca in the space community and is also usually spoken by engineers in non-English-speaking countries.

Upon reaching the survey starting page, potential participants were informed about the survey topic, background and organizers, structure, and duration. Anonymity was promised and participants were informed that their participation was voluntary and that they could quit at any time. Using web browser cookies, participants could later resume an interrupted survey if they desired to do so.

The questionnaire tool was open for a period of four months from November 2019 to February 2020. During this time, we continuously promoted the survey to attract participants to it.

2.4. Recruiting of Participants

Participants were recruited in four ways:

- Personal contacts as participants,
- Personal contacts as multipliers that emailed to their contacts and posted messages on social media platforms,
- Presentations and presence at two European conferences, an industry workshop on software for space, and an ECSS standardization group meeting,
- Providing a web-based form so that participants could easily invite other participants (using a mailto link),
- Paid advertisements in search engines and social media networks.

The personal contact channels ensured a sizable portion of participants from our target domain aerospace. Public advertising and personal contacts as multipliers added participants to the non-space control group.

In total, 305 responses were generated through paid advertisements, while 100 responses were attracted through one of the different personal channels. Respondents of the advertisement group had a 38% chance of completing the questionnaire, while 77% of the contacts group finished it.

As an accompanying measure to try and improve participation from the commercially advertised campaigns, a donation of 5 EUR to one of five charitable organizations of the participant's choice was promised for completing the questionnaire at the beginning of the survey. Moreover, participants who registered after the survey would be granted first access to a summary report that was compiled after the survey.

A total of 22,599 potential participants visited the survey landing page. Out of these, 4853 participants answered at least one question or proceeded to the first page beyond the landing page. This means that a remarkably high percentage of potential participants attracted to the survey did not proceed beyond the introduction page. Investigation showed that these participants were primarily attracted through the commercial advertising channels. We, therefore, experimented with different versions of the landing page to increase the "conversion" rate. Between different revisions, the content remained the same, but the presentation was changed to make the survey more appealing. However, while the discontinuation rate remained high, at least 196 completed questionnaires can be attributed to the commercial advertising services.

2.5. The Survey Instrument

Many of the question items must be seen in the context of a project. Therefore, after a few generic questions, participants were instructed to think of one of their recent projects and answer project-related questions in the context of this one project (called "the project").

Space projects are known for their complex supply chains that often span several tiers of customers and suppliers. A pivotal question, therefore, was whether a participant acted as a customer or supplier in the project and whether the software was developed for an organization's own purposes or projects. It was also possible to select "Other" and enter a text describing one's role. The option was selected 108 times in the full data set but there were no valid and complete responses using this option; all participants choosing this option came through search engine advertising. Analysis of the text descriptions showed that there were no relevant responses in the set. Participants could check multiple roles (e.g., an organization in the middle of a supply chain) but at least one role had to be selected.

The choice of supplier/customer role had a major influence on the rest of the questionnaire. Many questions were presented only if the appropriate role was selected. For example, if a participant did not act as a customer in the project, customer-related questions were not presented.

Furthermore, participants could skip most questions. There were only a few mandatory questions (e.g., the customer/supplier role questions).

Table 1 shows the structure of the survey instrument. Although quite similar, the order in which the topic and question items are shown here is not necessarily how they were presented to the respondents. Further details regarding the questionnaire are presented

in Appendix A which also shows how the respective data fields in the SKQuest data set are named.

Table 1. Overview of the SKQuest survey instrument and data.

Topic	Summary of Question Item and Answer Options
Introduction	The interview started with the background and organization of the survey (e.g., context, topic, duration, etc.), information regarding anonymity, and voluntariness of participation. The data set also contained metadata for each response including duration, completeness, whether the respondent was invited through a search engine, and a manual assessment of whether the response was valid.
Demographics	There were a couple of questions directed at understanding the participant's background and context: whether they were involved in a software project, the domain of work, company size, office location, experience with software and metrics, and development culture with respect to whether it is research or product development. For reasons of anonymization, office location and company size are not included in the published data set, and the domains are collapsed into aerospace and non-aerospace.
Project demographics	Upon reaching this part and before continuing with the questionnaire, participants were instructed to now think of one concrete current or past project. For the remainder of this survey, participants should answer project-related questions with respect to "the project". Information about the project gathered here included the participant's roles in the project, the degree of agility (as in agile software development) in project management, the project budget, whether participants acted as customers or suppliers in the project, and (if applicable) whether the customer was a public customer.
Status quo of project execution, product quality, and transparency	Several questions addressed the current situation in "the project" with respect to satisfaction with how the project ran, the quality of the product, and transparency. It asked about performed activities that increase transparency, and the use and technical reporting of software metrics.
The role of transparency for project success	This part of the survey addressed the role that transparency has in project success. It covers the expected importance of the development process for product quality, the various effects that transparency has on project success, whether transparency positively or negatively impacts project execution, and process and product quality.
Increasing transparency	Another complex of questions is whether increased transparency is acceptable to participants, and what activities are useful for increasing transparency. Additionally, a special focus with additional question items was provided on software metrics as a means of improving transparency.
The usefulness of and increasing transparency with metrics	This set of questions addressed the usefulness of metrics in general. It also inquired how often metrics should be used to improve transparency, and what cost for the metric gathering would be acceptable. Finally, it asked for a perspective on what kinds of metrics might be missing that could guide further research.
Quality of tools and support for metrics	The last part of the survey addressed the vision of supporting aerospace, industry, and research with a cross-institutional database of metrics, which kinds of future metrics are needed, and it captured opinions about the current tool landscape for gathering metrics.
Closing remarks	The survey concluded with closing remarks and the opportunity to provide a free-text comment regarding the survey or any other open points. The free-text comment is not included (see discussion below). As a thank you for their participation, participants were asked to which charitable organization we should donate (which is also not included).

2.6. The Underlying Quality Model

Some questions of the survey either directly (e.g., research directions for future metrics) or indirectly (e.g., quality characteristics associated with a metric) rely on a quality model. For example, the metrics defined in ECSS-Q-HB-80-04A [9] are associated with quality characteristics and subcharacteristics. Therefore, at least a roughly defined quality model is necessary as context for the survey. Note, we do not need or aim to define a consistent quality model. We needed terms for concepts that respondents could intuitively relate to.

The basis for our quality model was ECSS-Q-HB-80-04A. It defines a quality model consisting of the main characteristics of *functionality*, *reliability*, *maintainability*, *reusability*, *suitability for safety*, *security*, *usability*, and *software development effectiveness* (see [9]). The

quality model also defines subcharacteristics for each main characteristic. For example, portability as a member of the reusability main characteristic. However, our model neither relied on subcharacteristics nor did it use subcharacteristics from other sources.

Next, we complemented *software development effectiveness* with *software development efficiency* since both terms are often used together and denote distinct aspects. In the style of Peter Drucker, development efficiency means doing things right, i.e., the ratio of effort to output, whereas development effectiveness means doing the right things, i.e., reaching the objective.

Since ISO 25010 [12] (or its predecessor ISO 9126) is well-known to practitioners, we included the main characteristics of this standard in our model as well. This applies to *portability* (which, as mentioned above, ECSS lists as a subcharacteristic of reusability), *compatibility*, and *performance efficiency*.

In summary, our quality model consists of eight characteristics from ECSS, one to complement effectiveness, and three from ISO, making a total of 12 characteristics.

2.7. Overview of the Software Metrics in SKQuest

The SKQuest data set has a group of variables that contain an estimate of individual metric usefulness. The group of metrics was derived from ECSS-Q-HB-80-04A [9]. We chose this set of metrics out of practical reasons as practitioners, since the metrics from ECSS are specifically selected for use in European spaceflight projects.

On a general note, the ISO nomenclature has different terms such as base, direct, derived, or indirect measure. The ECSS simplifies this to just "metric", and there are two kinds: process and product metrics. Product metrics measure the products of development. Process metrics measure the development processes themselves [9].

Prototypical process metrics are the resulting maturity levels from a SPICE4Space Process Assessment or Milestone Tracking, as determined from the discrepancy between planned and achieved project milestones.

The product metrics include a subgroup of metrics that specifically target object-oriented programming. There are five metrics in this subgroup. A typical representative is the "Number of children" that helps to assess the complexity induced by inheritance. In our experience, safety-critical software is still often developed in C instead of an object-oriented programming language such as C++. Therefore, we did not include object-oriented metrics in our study.

The ECSS-Q-HB-80-04A defines 38 metrics (not counting the five object-oriented metrics). Seven out of these are process metrics, while the vast majority are product metrics. The descriptions of some metrics include variants. For example, the metric "Structural coverage" is defined as the ratio of code that was executed during the validation and verification phase. It is computed from executed and total statements, decisions, or conditions. These distinctions are our three variants.

The typical description of a metric in ECSS-Q-HB-80-04A contains the following information: a section number from the document serving as an identifier number (e.g., A.3.3.18), a short name (e.g., modular cohesion), the main and subcharacteristics from the quality model that the metric addresses (e.g., maintainability and modularity, respectively), a short description of the goal (e.g., "This metric provides an indication of the functions assigned to a single software module"), owner and producer of the metric (e.g., software product assurance manager and development team), target audience (e.g., software product assurance manager and development leader), the typical evaluation method (e.g., manual code analysis), a formula to compute the metric, interpretation guidelines for typical values, when to measure (e.g., during software design), applicability to criticality levels with respect to software safety and reliability, preconditions, recommended report format, and other remarks.

Since presenting the full description to respondents would take too much space and time, we decided to create short descriptions of the metrics. The full list of metrics and their descriptions is provided in Table A2 in Appendix B.

2.8. Data Filtering, Permutation, Correcting, and Amending

We excluded responses from the data set that were completed/aborted after less than 300 s, or that did not answer the first question. Note, since the first question was mandatory, this means no question was answered. Furthermore, the data set had been permutated randomly to not allow inferring participation order as this counteracts anonymization.

We manually judged all responses in the data set for plausibility. Responses, where the check failed, were flagged as bogus. Instead of removing these responses from the data set, we provided the bogus flag as a variable in the data. We recommend not using responses flagged as bogus, but users of the data set may come to different conclusions.

Our original design did not take the participant role of "student" into account. However, since "student" was given as a free-text response to the role question several times, we added a variable for the project role "student".

For anonymization reasons, some more detailed response data were collapsed into more general types. The twenty different answer options for where respondents worked were mapped to the two variables aerospace and non-aerospace. Furthermore, the country where their office was located, and the company size were removed. Moreover, since free-text responses may sometimes limit anonymity, and provide no forum for bogus messages, we also excluded 61 instances of free-text feedback. Most of the free-text responses had little informational value (e.g., "yes", no", "I don't know"), although sometimes cheering up (such as "thank you" or "very useful"). However, the comments helped us to identify bogus responses, for example, if the comment consisted of names of low-ranking local celebrities, statements claiming personal uniqueness, or mobile phone product names. Some of the valuable comments were integrated and preserved in this paper.

2.9. Related Work

A classical publication on software metrics is Basili et al. [13]. In their practice-driven research, software metrics assume a key role in assuring the quality of space projects. They postulate that the internet will reduce the troubles of collecting data, and this is where the AENEAS project starts. However, as our data shows, many of the problems related to software measurement are not rooted in the technical issues of transferring the data. Similar to the work of Basili et al., Prause et al. [2] describe the customer-side quality assurance (called product assurance) work of the German space agency from a practitioner's perspective. This paper must be considered in this context.

Software engineering literature knows many collections of metrics. Starting from classical sets such as Chidamber and Kemerer's object-oriented metrics [14] to systematic mapping studies such as the one by Saraiva et al. [15] with hundreds of metrics. Since so many metrics have been proposed in the past decades, there is even research on how to best catalog metrics [16]. Then again, databases such as PROMISE [17] provide software measurements from real-life projects. The SKQuest data set, instead, provides expert opinions about software metrics, not concrete software measurement results.

Vogel et al. [18] observed that partners in automotive supply chains lack a shared understanding of quality attributes and the right metrics to assess them. This leads to misunderstandings and costly renegotiations. To mitigate this problem, Vogel et al. describe a tool that helps users select a set of proper metrics for desired product qualities based on a big and precise metrics definitions database. In contrast to this, the ECSS already defines a sector-wide set of standard metrics. The SKQuest data set can help to understand better opinions about this set of metrics and optimize it and the standard.

The vision of many software metrics researchers is to use them for defect prediction. Son et al. [19] present a systematic mapping study on defect prediction methods. Research usually focuses on code metrics. With its choice of metrics, the SKQuest data set can be considered to provide metrics for risk reduction in business relationships.

Several software measurement tools such as Squore are available as commercial software to support software measurements. Additionally, there is still research going on to develop new metric tools. For example, Choras et al. [20] report from the recent Q-Rapids

project. The project developed a software measurement tool and evaluated it for improving processes in small companies. As opposed to these efforts, the SKQuest data set—and the AENEAS infrastructure—aim at improving transparency across company boundaries.

Choras et al. further notes that research on software metrics has a long history. Yet, they find that literature on measurement and agile development is scarce and that the literature on process improvement for small or medium enterprises is even scarcer [20]. The New Space movement (cf. [21]) emphasizes the need to better understand the role of metrics in private and agile spaceflight settings. The SKQuest data set supplies data to conduct these kinds of research.

The systematic literature review by Ofem et al. found only 18 relevant publications that deal with transparency in software engineering in a period from 2006 to 2022. They concluded that transparency is an emerging concept in software engineering that is often neglected [22].

Saraiva et al. discuss and statistically investigate the typical limitation of software measurements programs, i.e., wasting effort due to inconclusive and erroneous data analysis based on useless, redundant, incomplete, or low-quality data, or concisely stated in the dictum "garbage-in, garbage-out" [23]. Perceptions of the usefulness of software measurements can also be analyzed with the SKQuest data set, and this paper provides a new view of this question.

This paper provides a data set to allow research into the circumstances and contexts where software measurement works. An important aspect of the data is transparency, but it is by far not limited to this aspect. The authors are not aware of a publicly available data set on software measurement that is comparably extensive, both in terms of the number of responses as well as variables.

3. Results

This section describes the SKQuest data set. It first provides an overview of the data set through key figures. Next, it gives answers to the primordial questions defined earlier, and tries to replicate results from Prause and Hönle [6] with this bigger data set.

The SKQuest data set can be downloaded as csv file from the web address provided in the supplementary materials section at the end of this paper.

3.1. Key Figures of the Data Set

The data set contains a total of 405 responses across 190 variables. Out of these responses, 211 were incomplete, while 194 were complete. A total of 305 responses were generated through advertising on social media and search engines, while the remaining 101 originated from direct and indirect personal invitations. A total of 174 incomplete and 114 complete responses were manually judged as valid responses (i.e., flag $mBogus set to false). Unless noted otherwise, this paper reports results only based on responses that were judged as valid.

While there is a simple criterion for whether a response is completed (i.e., reaching the final page), the degree of completeness of incomplete responses is more difficult to determine. The reason is that not all questions were mandatory, and some questions may never have been presented to a participant depending on earlier responses (e.g., questions targeted at suppliers if the participant was a customer). Therefore, the data set contains the $mCompleteness variable, which is based on the number of logically coherent sets of questions that were considered completed when at least one question of the set was answered. Based on this definition of completeness, completed responses were 95% complete on average; incomplete responses were 32% complete on average.

Out of the complete responses, 32 were customers with 19 of them being customers exclusively, i.e., not acting in one of the other two roles. A total of 63 responses were from suppliers including 42 responses from exclusive suppliers. In addition, 46 responses came from respondents involved in development for internal purposes, including 31 exclusively internal developers. A total of 35% of complete responses came from participants working

at public institutions, while 51% worked for private companies (the remainder provided no answer).

A total of 109 respondents (including incomplete responses) worked in the space and aviation domains, and 168 (also) worked in other domains. A total of 198 respondents did not answer this question. A total of 130 respondents had ten or fewer years of experience in their domains, while 74 had more than 10 years, amounting to an estimated total of almost 1500 years of domain experience. A total of 132 respondents had high or very high experience in software development, and another 42 respondents had at least a medium level of experience, while 29 had low or very low experience. Regarding respondents' experience with software metrics, the situation was similar: A total of 131 respondents felt that they had the same experience with software metrics as with software development. However, 52 respondents had less experience with metrics than with development, as opposed to 20 respondents that had more experience with metrics.

The vast majority (67%) of responses originated from Europe. The reasons for this were, of course, the focus on European spaceflight and its ECSS standards, and respective personal contacts. A total of 22% of responses came from Asian countries, 9% from Southern (8%), and Northern America (1%). African respondents contributed 3%.

3.2. Answering the Four Primordial Questions

We defined four primordial questions that drove the creation process of the survey instrument; however, they were not defined sharply enough so that we would not consider them as real research questions. Instead, we used them as a means of showing what could be done with the data in the SKQuest data set.

Correlation coefficients express the strength of some correlation. As a preliminary remark to the following discussion, please note that it is difficult to state what constitutes a "strong" correlation. In social sciences, where people and opinions are involved, outcomes often depend on many input variables. Therefore, correlation coefficients are usually lower when compared to, for example, their counterparts in physical sciences which are based on a few natural laws. We orient toward the rules of thumb of de Vaus [24].

3.2.1. Question 1: Is Transparency a Problem in Software Development Projects?

The data are ambivalent with respect to this question. On the one hand, respondents felt well informed but, on the other hand, they agreed that there is a lack of transparency and that more transparency would benefit their projects. Before we come to these results, it is important to understand how the different expressions of satisfaction with transparency relate to one another.

Table 2 relates the four variables that measure respondents' perceived transparency in their project, i.e., feeling well informed about the project and product, the occurrence of surprises, and the perceived over-informedness of non-developing outsiders. The table shows correlations for each group of respondents for customer, supplier, and internal development. First, we noted strong positive correlations for the two feeling-well-informed variables; however, we also see that the negatively connotated variables (occurrence of surprises and over-informedness) also positively correlated with feeling well informed, except for customers. For customers, surprises might be associated a little bit with feeling not-so-well-informed. Customers have a different point of view compared to the other two groups because customers are the group who receive information from an external development effort, whereas the other two groups develop and "export" visibility of the project and product.

Now, to understand whether "transparency is a problem", we established a fundamental relationship between development success and transparency; see Table 3. Project success is measured through satisfaction with product quality and process compliance and efficiency. Perceived transparency is measured through the feelings of being well informed, the perceived occurrence of surprises, and feelings that non-developing parties (i.e., customers or internal stakeholders) are over-informed.

Table 2. Pearson correlation coefficients for the interrelationship between satisfaction level and transparency variables, i.e., feeling well informed about project status ($s[c/s/o]InfoStatus) and software quality ($s[c/s/o]InfoQuality), the occurrence of surprises ($s[c/s/o]Surprises), and over-informedness ($s[c/s/o]Overinfo). The coefficients in each table cell are for customers/suppliers/internal developers. Correlations that are not statistically significant (i.e., $p \geq 0.05$) are marked with ⚠.

Interrelations for the Status Quo of Transparency Variables	Well Informed about Software Quality	Occurrence of Surprises	Development Outsiders Are Over-Informed
Well informed about project status	0.67/0.75/0.74	−0.10 ⚠/0.42/0.32	0.38/0.20/0.24
Well informed about software quality		−0.03 ⚠/0.40/0.48	0.51/0.30/0.39
Occurrence of surprises			0.16 ⚠/0.50/0.39

Table 3. The relationship as Pearson correlation coefficients between transparency (as expressed by feeling well informed about project status and software quality, the occurrence of surprises, and over-informedness) and satisfaction with the software product and its development process from the customers'/suppliers'/internal development teams' point of view. Correlations that are not statistically significant (i.e., $p \geq 0.05$) are marked with ⚠.

Satisfied with …	Well Informed about Project Status	Well Informed about Software Quality	Occurrence of Surprises	Development Outsiders Are Over-Informed
Product quality	0.61/0.47/0.63	0.59/0.50/0.56	−0.04 ⚠/0.20/0.26	0.14 ⚠/0.30/0.54
Process compliance	0.66/0.63/0.53	0.70/0.61/0.48	−0.04 ⚠/0.42/0.29	0.41/0.26/0.60
Process efficiency	0.58/0.58/0.54	0.61/0.60/0.59	0.16 ⚠/0.36/0.31	0.48/0.35/0.58

Satisfaction with the product and its development process correlates strongly with transparency regarding project status and product quality (the first two 3 × 3 blocks). Hence, transparency has a strong impact on project success, and transparency is a problem particularly if it is not there. As far as suppliers are concerned, the data clearly advise them to let customers feel that they are well informed.

In the third block, we did not find evidence that the occurrence of surprises is (negatively) related to satisfaction among customers. The occurrence of surprises might indicate that the supplier has insufficient control over the project, which might be perceived as a bad thing. However, there is no significant linear correlation. At the very least, this could mean that it is not bad to handle surprises openly. For suppliers and internal development, interestingly, there is a low to medium positive correlation between surprises and satisfaction. It could mean that these groups, being closer to development than customers, perceive surprises as a normal part of the development and that, therefore, a culture that deals with surprises openly is, in fact, seen as a good thing.

Lastly, the fourth block shows that customers who feel over-informed tend to be satisfied with process compliance and efficiency. There might also be a weak tendency to be satisfied with product quality; however, this effect is not significant. Suppliers, on the other hand, who think that their customers know too much, tend to be satisfied with product quality as well as the compliance and efficiency of their development process. The same effect, only much stronger, also occurs in internal development with respect to stakeholders from inside the same organization.

Much in the same vein, respondents considered transparency as directly beneficial: They expected a positive effect of transparency on processes, the product, and project execution (see Figure 1). About 50% of respondents stated that transparency makes process, product, and project execution much better, and another 25% said that it makes it at least slightly better. Contrariwise, respondents that saw a negative effect of transparency were few and far between and accounted for barely more than 10% of respondents.

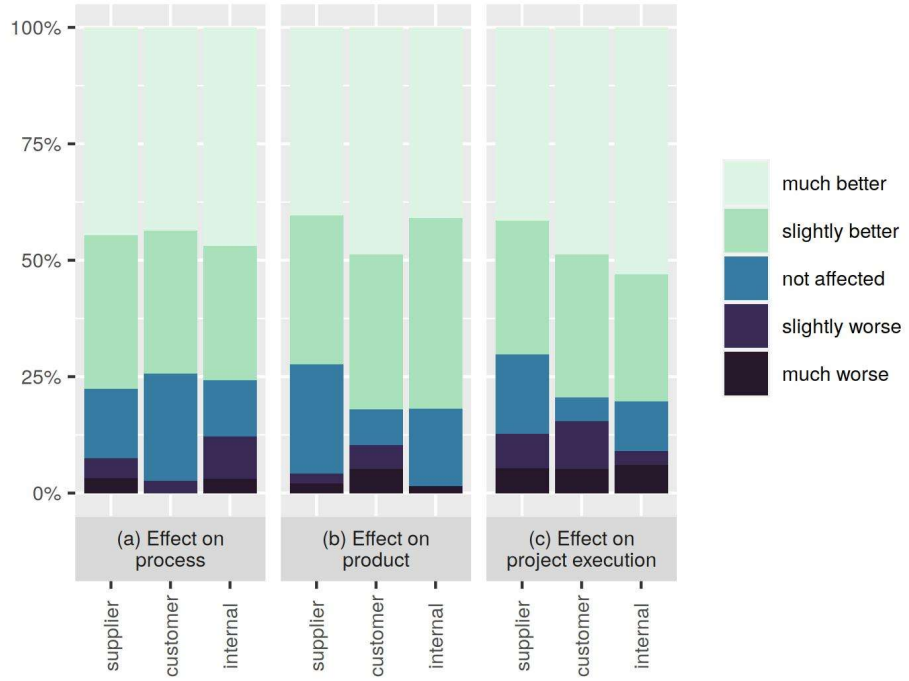

Figure 1. The expected effect of transparency on process, product, and project execution for suppliers, customers, and internal development projects if transparency increases compared to its current state, as obtained from variables $tEff[C/S/O]Proc, $tEff[C/S/O]Prod, and $tEff[C/S/O]Proj.

Incidentally, we note that respondents considered the process as especially important for the quality of the product. Given an interval from 0.0 (no relevance of process for product quality) to 1.0 (extremely important), respondents rated the importance of the process for product quality at 0.76 on average. This number was no surprise because the importance of the process has long been recognized. However, few numerical expressions of the relationship exist.

Of course, the question of whether software development has a problem with transparency is strikingly reflected in the statement about the emperor's new clothes (see Table 4). Two-thirds of respondents who answered this question agreed that there is a huge problem with transparency in software development. (Remark: Please note the small number of responses since this question was introduced later and, therefore, presented only to a small portion of the overall participants). Looking at the two groups of respondents from public vs. private entities, private entities are a bit more concerned about opacity than public entities. An explanation could be that public entities tend to base their contracts on more elaborate specifications and that more transparency also means more involvement with respect to legal liability and personnel effort (cf. [21]).

As a side note, the Matthews correlation between being a private entity and a (rather) agile project execution ($agility > 3) is r = −0.21 ($p < 0.05$), i.e., members of private entities in our survey reported to have less agile projects. This finding might appear opposed to the hypothesis that public entities with their rigid and bureaucratic structures have more difficulties becoming agile than others (see discussion and references in [21]). However, agility is clearly linked to a more "researchy" culture (Pearson r = 0.39), and private entities in our study are very slightly more likely to have a bit less research-oriented culture (r = 0.18 ⚠, not significant).

We also tested for a correlation between the transparency variables (feeling well informed about project status and software quality, the occurrence of surprises, and over-informedness) and agreement with the emperor's-new-clothes-statement (see Table 5). Customers who felt well informed about the software and its development tended to agree that transparency is a problem. For respondents who have been involved in development themselves (either as suppliers or in internal development projects), the occurrence of

surprises correlates with an agreement to transparency problems. Customers and developing respondents all tend to see transparency as a problem when they also feel that non-developing parties know too much. It is important to note that feeling over and well informed correlate positively.

Table 4. Agreement with the statement that software developers are too much the weavers in the story about the emperor's new clothes. The table shows overall numbers and numbers in selected subgroups. The low response counts are due to missing values in the emperor's-new-clothes-question.

	Fully Disagree	Disagree	Neutral	Agree	Fully Agree	Mean [−2, 2]
Overall	2	3	7	9	15	0.89
(fraction)	6%	8%	19%	25%	42%	-
Public entities	1	2	4	5	8	0.85
(fraction)	5%	10%	20%	25%	40%	-
Private entities	1	1	2	4	6	0.93
(fraction)	7%	7%	14%	29%	43%	-
Fully agile or agile projects	0	1	1	2	6	1.30
(fraction)	0%	10%	10%	20%	60%	-

Table 5. Pearson correlation coefficients between an agreement with the emperor's-new-clothes-statement and transparency, as perceived by customers, suppliers, and in internal development. Only $|r| > 0.2$ values are shown. ⚠ marks statistically not significant results.

	Well Informed about Project Status	Well Informed about Software Quality	Occurrence of Surprises	Development Outsiders Are Over-Informed
Customer	0.46 ⚠	0.74	-	0.66 ⚠
Supplier	-	-	0.54	0.45
Internal development	-	-	0.55	0.40 ⚠

3.2.2. Question 2: Is There a Desire for More Transparency in Projects?

A willingness to accept more transparency is not strictly the same thing as wishing for it. But the two statements are similar. We, therefore, use willingness to accept more transparency as a proxy for wishing for more transparency. We asked respondents in the roles of suppliers or internal development projects whether they would be willing to accept increased transparency of their processes and the current state of software development ($accIncT). In response, 3% of respondents fully rejected to accept increasing transparency, another 8% rather rejected it, and 16% of respondents were indifferent. However, most respondents would rather (30%) or fully (42%) agree to increase transparency (n = 128). We also found a correlation of r = 0.42 between willingness to accept increased transparency and agreement with the emperor's-new-clothes statement. The intuition is that someone who sees problems with too little transparency is also willing to accept more transparency.

For a different angle on the question, we looked at the variables from the block "satisfaction with transparency" (see Figure 2). Most project participants felt well informed regarding project status and product quality, even though surprises seem to happen from time to time. The fourth triplet captures whether (i) customers felt they knew too much about the project, (ii) suppliers felt that customers knew too much, and (iii) other parties inside the same organization had too much insight. While there is less agreement than with the other statements, still, many participants felt that customers or other internal stakeholders knew too much.

However, we cannot explain the correlation of r = 0.74 between agreement with the emperor statement and feeling well informed about product quality as a customer, i.e., while some respondents were worried that software development lacks transparency, they still feel well informed in their own projects.

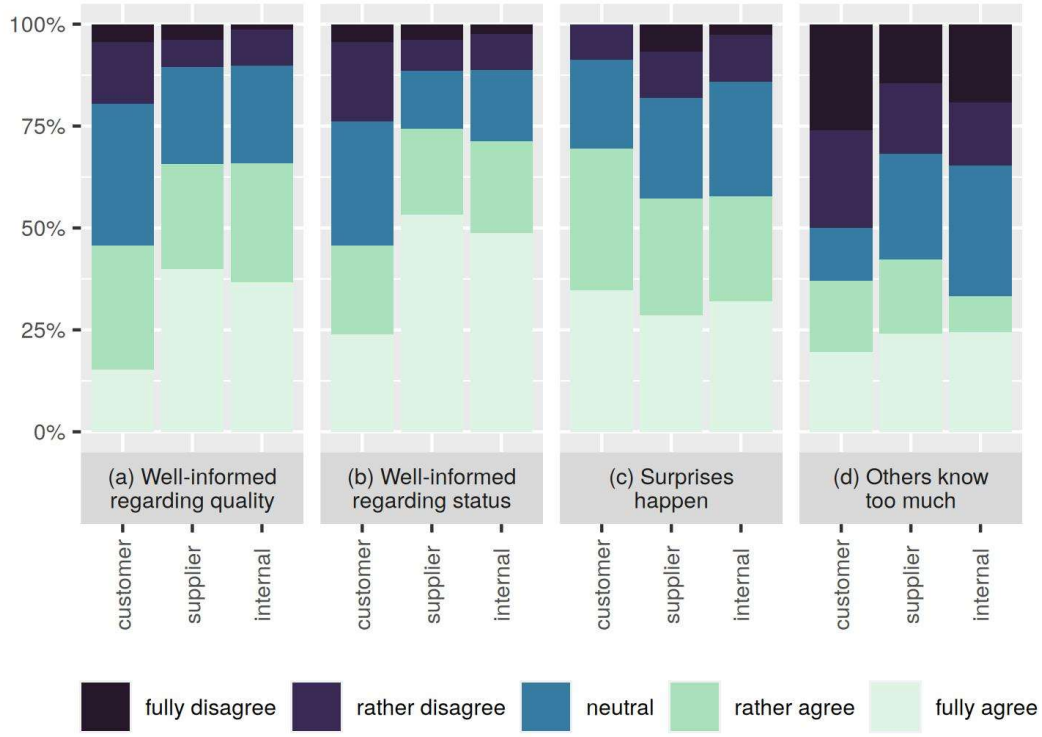

Figure 2. Satisfaction with the status quo of transparency, each captured by a triplet of separate customer, supplier, and internal views. The four topics are feeling well informed regarding software quality ($s[s/c/o]InfoQuality) and project status ($s[s/c/o]InfoStatus), the occurrence of surprises in project execution ($s[s/c/o]Surprises), and feeling that others know too about the project ($s[s/c/o]Overinfo).

Comparing the bars within each triplet, the data shows that customers feel less well informed about the quality of the software they procure than suppliers and internal development teams about the software they develop themselves. This, of course, comes as no surprise, since there is no organizational border that could limit the flow of information. The same holds true for the status quo of development. Vice versa, a remarkable observation is that customers felt that surprises happen more often, although they are farther away from development, and, therefore, not all tidings of surprises reach them because they are filtered out before reaching them. Hence, the remaining surprises weigh heavier. This effect has the same origin: it means that customers are less well informed about project execution.

The fourth triplet reveals a different pattern. While customers do still feel that they receive less information, suppliers particularly think that customers know too much. Internal development teams are more willing to accept that information about the project reaches other parties inside the same organization.

Transparency can have desirable as well as potentially undesirable effects. Figure 3 shows that while respondents recognize dangers in potentially unwanted effects, the desirable effects are more prevalent. In addition, note that some of the potentially undesirable effects might, in fact, be wanted.

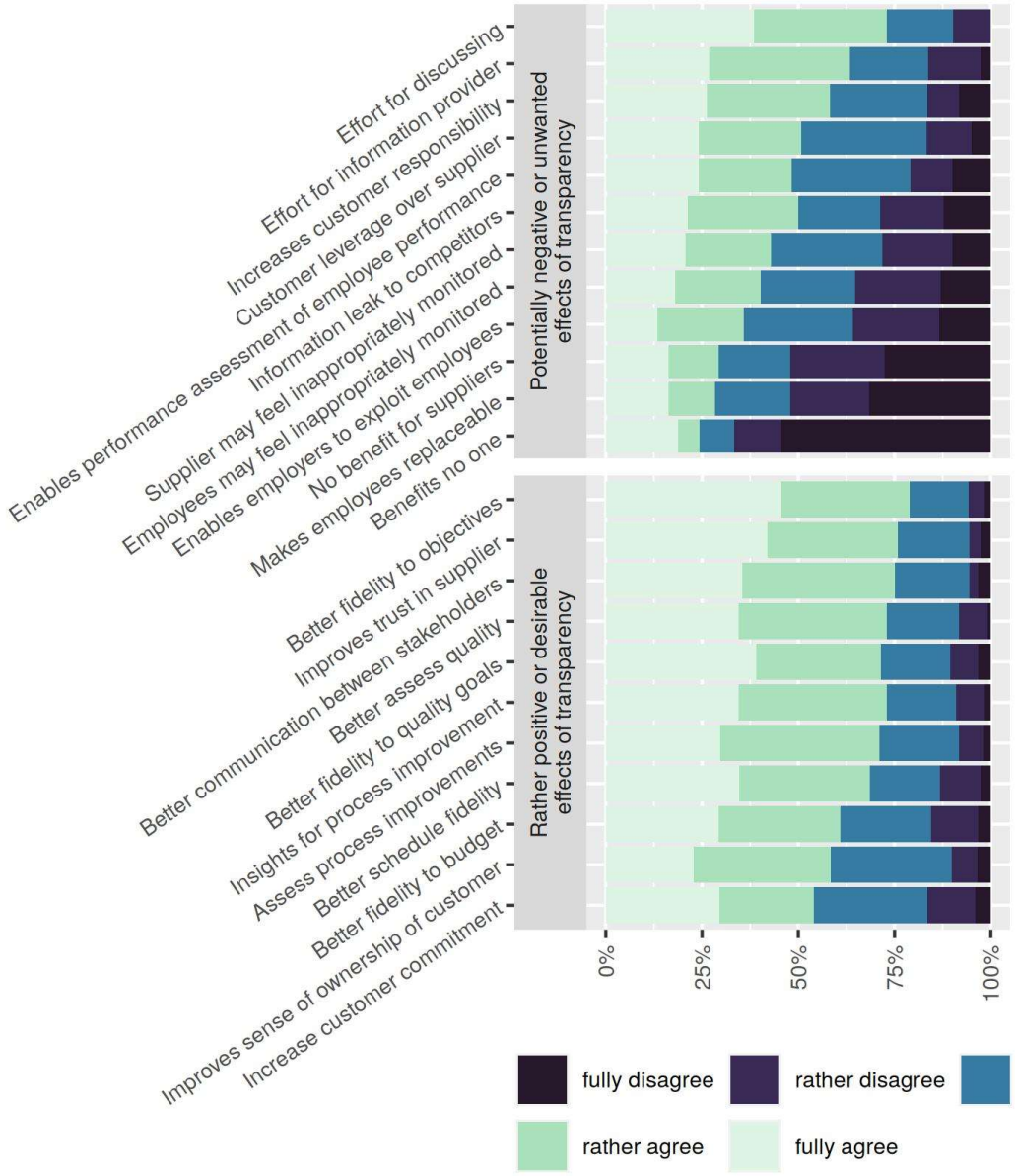

Figure 3. Desirable vs. potentially undesirable effects of increased transparency. We use "probably" and "potentially" here because whether the effect is desired or undesired also depends on the culture and whether one is a customer or supplier. The distinction between good and bad is not sharp.

3.2.3. Question 3: Can Metrics Contribute to Improving the Situation?

While this is not a direct answer to the question, respondents would be willing to invest part of the project budget in measurement. On average, respondents would accept investing 9.6% of the annual project budget for regularly gathering metrics ($accMetCost). Customers would even accept 10.4% for measurements, while suppliers would accept only 8.6%. The least investment of 8% can be expected from internal development projects. After all, the difference is not so noticeable. Moreover, it appears to be related to the fact that customers feel a bit less well informed about project status and product quality than suppliers. In addition, the data might show correlations between acceptable cost and agreement to the emperor's new clothes statement ($r = 0.21$ ⚠), customers' informedness regarding project status ($r = 0.21$ ⚠), and product quality ($r = 0.26$ ⚠). Note that the correlation coefficients are weak to moderate but not statistically significant, which means that there is a risk of over 5% that we observed them by chance. We did not find analogous correlations between suppliers and internal development.

Figure 4 shows which activities respondents use in their projects to increase transparency, and how this affects satisfaction with the developed software. Overall, most respondents are satisfied with the software product. The most-used transparency activities are releases of intermediate software versions, meetings between technical personnel, and detailed documentation. While metrics are used by over 50% of respondents, they are rarely communicated and delivered. However, note that the list of activities is probably not exhaustive, and which transparency activities can be used in a development situation is highly context-dependent. Direct comparisons, therefore, have only limited expressiveness.

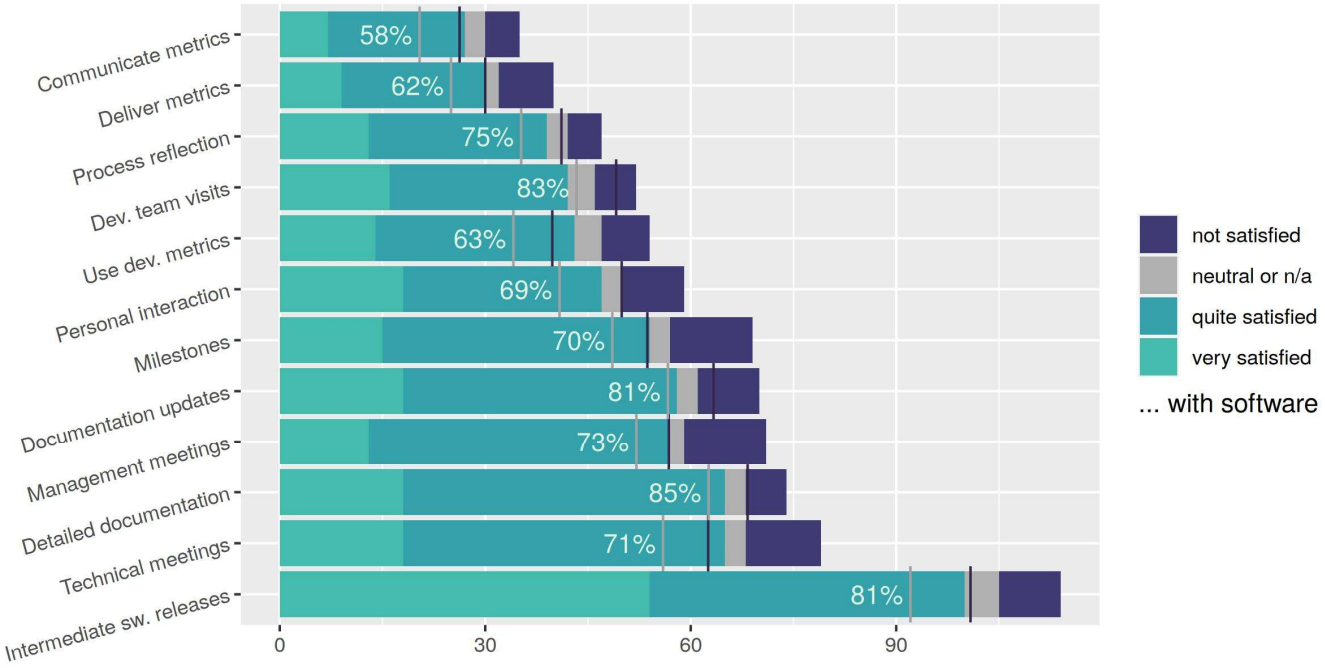

Figure 4. Activities that projects use to increase transparency are ordered by an absolute number of mentions. The colors show the degree of satisfaction with the developed software (per-response mean of $s[s/c/o]Software). The vertical gray line and the percentage indicate the relative value of the subgroup "only customers". The violet, second line represents the "not satisfied" mark for customers.

For example, the degree of agility of development has a strong influence on which transparency measures are used. Agile projects will make more use of intermediate releases (Pearson correlation r = 0.20) but less use of detailed documentation (r = −0.26) and updates (r = −0.20), delivering metrics (r = −0.21), milestones (r = −0.20), and technical meetings (r = −0.17). Most of this is congruent with the agile ideals of interaction, collaboration, and working software but, surprisingly, technical meetings are also avoided.

While the bars include all responses, the vertical lines show satisfaction for customers only. Very often, customers are less satisfied with the software than the average respondent. With respect to concrete transparency activities, visits to the development team (83%) and frequently detailed documentation (85%) seem to be particularly welcomed. Metrics are the least associated with satisfactory development results.

As opposed to this, when asked directly, respondents agreed that transparency increases when metrics are used (see Table 6). The values are on range from −2 (fully disagree) to +2 (fully agree). In general, more frequent delivery (+0.99) obtains more approval than increased volume (+0.82). Transparency gains from receiving software measurement data are expected for customers (+0.81 and +0.63 on average for increased frequency and volume, respectively). However, others within the same organization (+1.05/+0.88) and, in particular, the software team itself (+1.24/+1.05) are expected to benefit more. As opposed to this, customers themselves (+1.00) expect more benefits from metrics than suppliers (+0.94) or the average respondent (+0.91).

Table 6. Agreement to the statement that transparency increases for the customer, the development team, or other stakeholders within the developing organization if more metrics are obtained ($tIncAMet[C/S/O]More) or more often ($tIncAMet[C/S/O]Freq). Values range from −2 (fully disagree) to +2 (fully agree). The lower two rows present average responses given by customers and suppliers.

	Increased Frequency	**More Metrics**	**Mean**
For the customer	+0.81	+0.63	+0.72
For the software team itself	+1.24	+1.05	+1.14
For others within dev. organization	+1.05	+0.88	+0.96
The mean of rows above	+0.99	+0.82	+0.91
Customers' opinions	+1.13	+0.89	+1.00
Suppliers' opinions	+1.03	+0.86	+0.94

The expectation that metrics can improve the situation is expressed not only in the wish for more frequent delivery or voluminous data. Figure 5 depicts the perceived usefulness of different metrics from ECSS-Q-HB-80-04A [9] ordered by the mean value. Since only a randomly selected subset of ten metrics was presented to each respondent, the number n of responses per metric varies between 24 and 52. The number in brackets after the metric name is the section number in ECSS-Q-HB-80-04A that describes the metric. Some of the metrics in ECSS have variants that are listed separately. The variant is then named in square brackets "[var. …]".

In general, none of the ECSS metrics are rated as not being useful. Instead, all metrics are (on average) perceived as being useful to differing degrees. In fact, the perceived usefulness of metrics varies quite heavily: the horizontal lines show a standard deviation range of only 0.3σ, i.e., only 25% of responses can be expected to lie inside this interval.

While the figure shows metrics ordered by overall mean value, the symbols indicate the mean values of different subgroups of respondents such as customers, suppliers, agile development, and participants in aerospace projects. On the one hand, the different subgroups of respondents sometimes seem to come with quite varying expectations of the usefulness of the metrics, as indicated by chaotic symbol patterns. On the other hand, however, there still seems to be some deeper consensus across disciplines on what the more useful metrics are. This can be observed graphically in the figure, and mathematically by correlating the rankings of usefulness by different subgroups (Spearman rank coefficient ρ), for example: $\rho_{\triangle\triangledown} = 0.50$, $\rho_{\leftrightarrow} = 0.54$, $\rho_{\leftrightarrow} = 0.73$, or $\rho_{\leftrightarrow} = 0.82$. The comparison $\rho_{\leftrightarrow} < \rho_{\leftrightarrow}$ might indicate that respondents from aerospace preferred the same metrics as respondents from non-agile projects, although we found no direct correlation between project agility and working in aerospace. Note that most compared subgroups were not completely disjointed, e.g., individual respondents may have been customers and suppliers, or worked in agile and space projects, at the same time. This, of course, causes higher correlation coefficients.

One respondent noted that "There is some measure of transparency within my project and with my client, the problem is whether I can influence solving problems found due to schedule/budget issues". Therefore, there is a conflict between transparency and the need to finish on schedule and budget. Here, metrics could help because the process might be automated to some degree. Of course, this issue is not yet solved. Another respondent pointed in the same direction: "The metrics shall be collected (ideally also analyzed) automatically, otherwise the activity is too costly, too error-prone, and too repetitive, becoming a burden for team members who will try to avoid it".

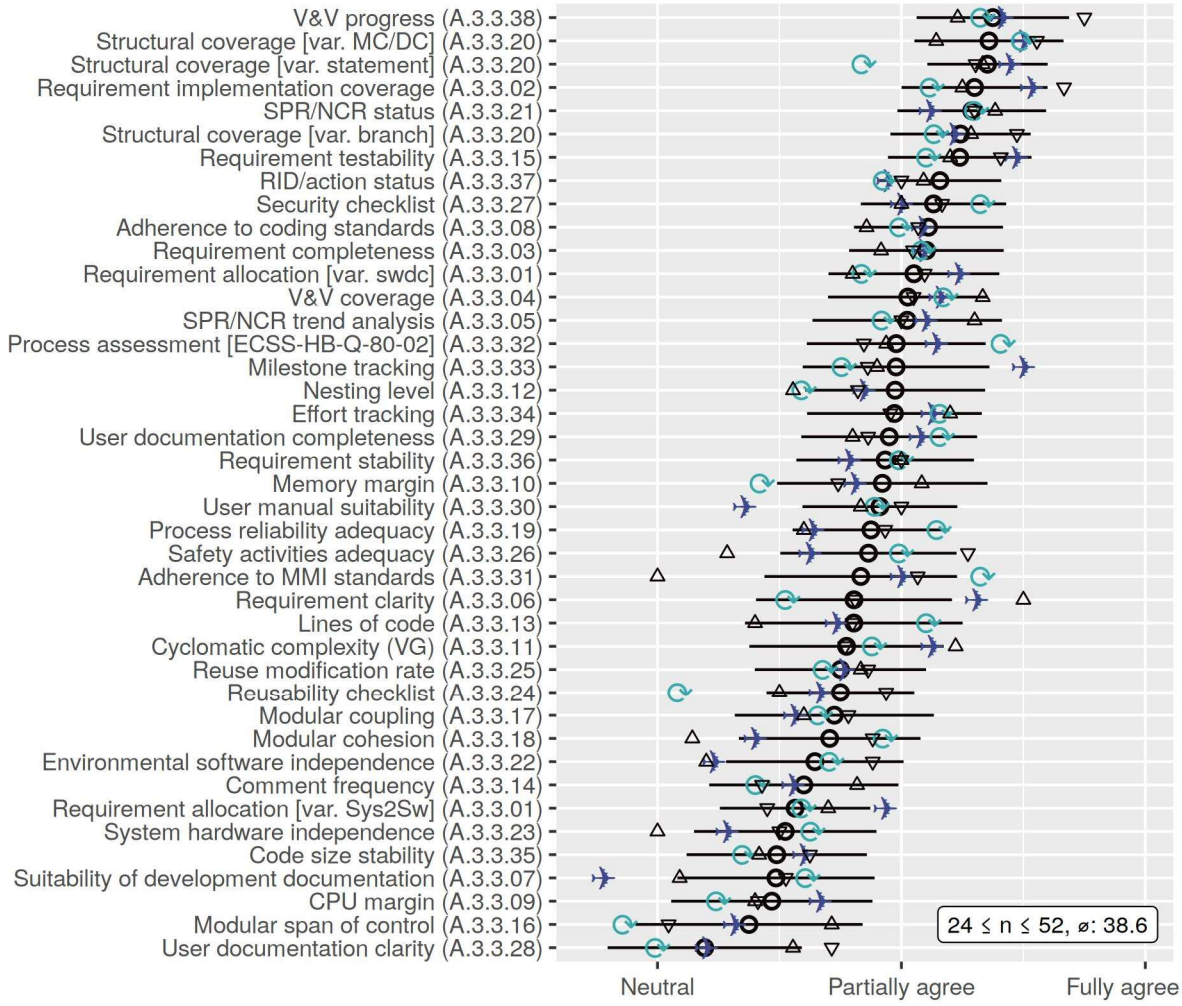

Figure 5. Perceived usefulness of the different metrics from ECSS-Q-HB-80-04A [9] ordered by their mean value (from −2 (fully disagree) to +2 (fully agree)). The black circles show the mean value, while the range indicates a one-third sigma standard deviation. The means for various subgroups are marked with △ (customer), ▽ (supplier), ⟳ (agile or rather agile development), and ✈ (aerospace domain).

3.2.4. Question 4: How Can AENEAS Fit into the Current Tool Landscape?

This question motivated the SKQuest survey, and several variables address this aspect. However, a too-narrow discussion of this question has extremely limited value for readers of this paper. One participant noted: "I have the feeling that there is a great focus put on the data interchange formats for metrics, the existence of tools, and the frequent providing of metrics […]". Therefore, we interpret this question more freely. The data then offers interesting insights into the current software measurement ecosystem.

Figure 6 presents the recommended reporting frequency of metrics. An interesting observation is that respondents tend to favor extremes, i.e., reporting in real time and reporting only very seldom. (Note: The answer option with the least frequency was "Up to once every three months". Hence, respondents might prefer even less frequent reporting). Reporting frequencies of "daily" and "weekly" are preferred by only very few respondents. Suppliers prefer less frequent reporting of metrics, even a bit more so than customers (means of 2.5 and 2.7, respectively). Both are, however, rather on the opposite side of respondents who are involved in the development of software for their organizations' own purposes (mean value 3.4). The latter tend to prefer more frequent reporting of metrics. In this group, "real time" is—together with "quarterly"—the most often selected option. While for customers and suppliers, the degree of agility of development in the

project does not seem to correlate with the reporting frequency, and we find a significant negative correlation of r = −0.37 for internal projects, i.e., agile development seems to conflict with frequent reporting of metrics. This finding appears counterintuitive since agile development favors transparency, which metrics can improve. Then again, the agile manifesto demands "Individuals and interactions over processes and tools", so that other means of creating transparency are more relevant.

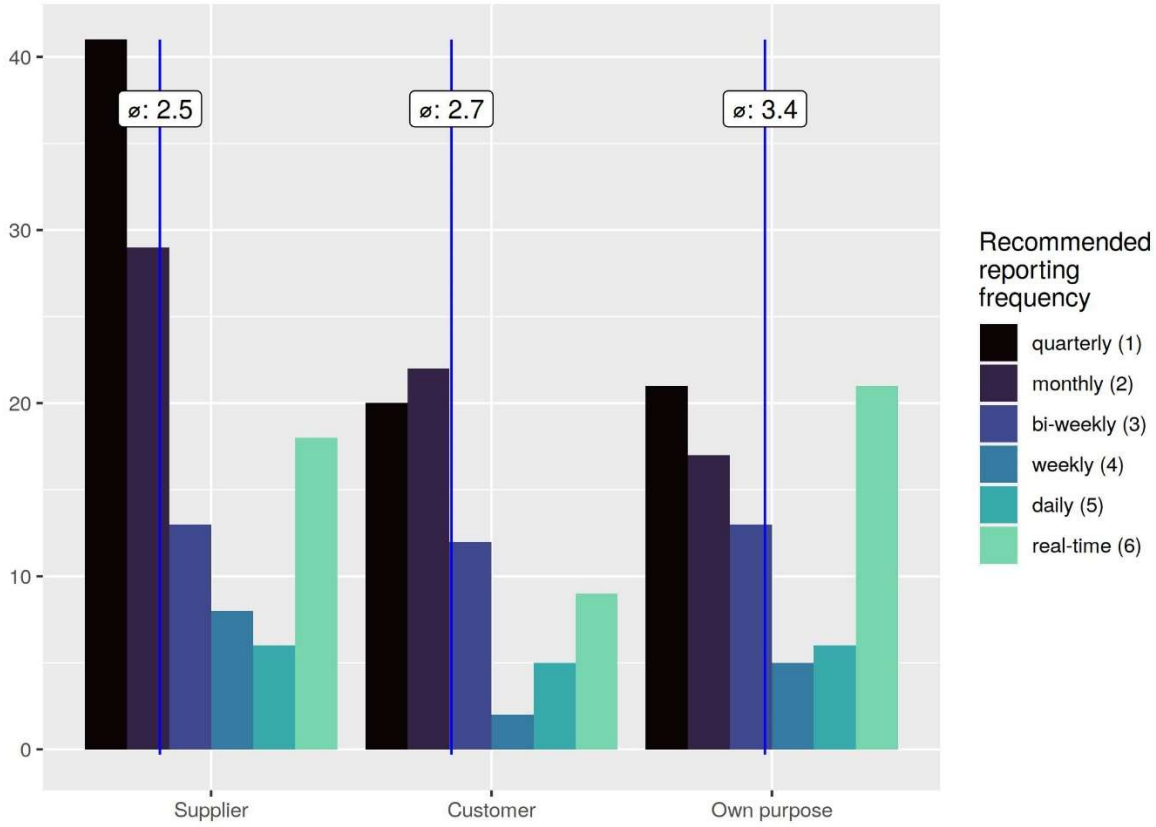

Figure 6. Recommended reporting frequency of metrics ($metRecFreq[S/C/O]). The blue lines indicate the mean of the respective group.

Figure 7 summarizes the responses to the question of what quality characteristic future still-to-be-developed metrics should target. Both process metrics rank in the mid-tier of need. Interestingly, the three characteristics from ISO that ECSS-Q-HB-80-04A did not include (compatibility, portability, and performance efficiency; see Section 2) finish last. Hence, the ECSS quality model might not just have "forgotten" these three characteristics but has potentially left them out intentionally because the committee's experts did not consider them as important or interesting.

The dashed red line shows results only for respondents working in the aerospace domain. Among them, reliability would be the most desired target for future metrics. This might be well reflected in the aerospace culture that emphasizes the reliability of systems. On the other hand, characteristics functionality and development efficiency might receive less attention from aerospace people than from all.

In a free-text response, one participant noted that "[…] if there is no improvement in the definition of metrics that can be automatically gathered, we will have an increasing gap between those metrics that are easy (e.g., we can measure LOCs automatically and provide them every day) and those that are difficult (e.g., is the software requirements document unambiguous?). In my opinion 'easy' metrics that can be integrated into tools and 'software factories' are not necessarily the best or important ones improving the quality". Hence, it is also important to look for smart metrics that are not so easy to measure.

Another respondent noted that many metrics are too technical for quality assurance and management staff who are often not software developers. They need more easy-to-understand metrics and more training/documentation on existing metrics.

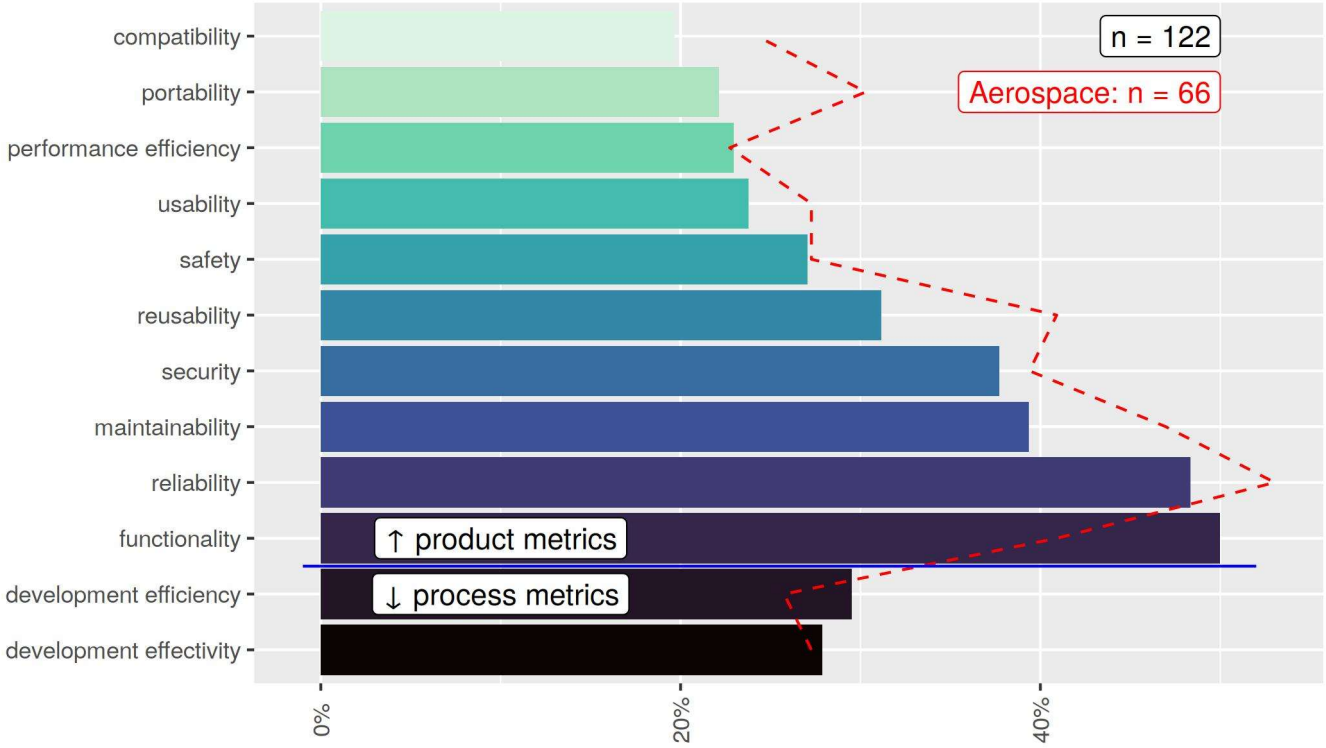

Figure 7. Areas that future still-to-be-developed metrics should target. The red dashed line shows the responses only from participants working on aerospace projects.

3.3. Replication of Earlier Results

As a second demonstration of the SKQuest data, we seek to replicate the findings of Prause and Hönle [6], a small interview study with spaceflight practitioners (n = 23). Besides demographic questions, it formulated the following three attitude questions:

1. (Q_i) Do you agree with the statement about software and the emperor's new clothes?
2. (Q_{ii}) Do you wish for more transparency in software development?
3. (Q_{iii}) Would regular delivery of ECSS metrics help you fulfill your role?

3.3.1. Mapping of the Attitude Variables

To replicate said earlier results, we must map the original questions to variables available in the SKQuest data set. While, evidently, there is a similarity to the three primordial questions, the mapping is not perfectly straight forward but can be performed based on the discussion in Section 3.2.

(Q_i) The SKQuest data set contains a variable from an almost identical question ($agreeEmp). Sadly, though, the number of available responses to this question was small (n = 36). Therefore, we define Q_i as the mean value of the three input variables Q_{ia}, Q_{ib}, and Q_{ic}. Q_{ia} is $agreeEmp. Q_{ib} is the mean value of the perceived occurrence of surprises (variables $s[c/s/o]Surprises). The question is similar to the emperor's-new-clothes statement, which describes the fear of an unpleasant surprise that the project runs very badly. Q_{ic} is the mean value of the expected effect that transparency has on project success (variables $tEff[S/C/O]Proj), i.e., whether more transparency leads to worse or better project success.

(Q_{ii}) While Primordial Question 2 is the same as Q_{ii}, there is no direct corresponding variable in the SKQuest data. Therefore, we define Q_{ii} based on Q_{iia} and Q_{iib}, in analogy to Q_i. For Q_{iia}, we use satisfaction with the status quo of transparency. However, we only use

the feeling that the external party is over-informed (mean of $s[s/c/o]Overinfo) because responses to this variable are more balanced than other variables in this group. Remember that over-informedness is positively correlated with feeling informed. For Q_{iib} we use the acceptability of increased transparency.

(Q_{iii}) This question finds its counterpart in Q_{iiia} and Q_{iiib}. Q_{iiia} is the expectation that an increased frequency of delivering metrics (cf. " … regular delivery of ECSS metrics … ") is beneficial (mean of $tIncAMet[C/S/O]Freq). Q_{iiib} is the mean rating of the usefulness of all metrics rated by the respondent ($metric[…]).

Figure 8 plots the results for the three variables Q_i, Q_{ii}, and Q_{iii}, as obtained from the original study's data [6] and the SKQuest data set. While there are some differences, the results are similar.

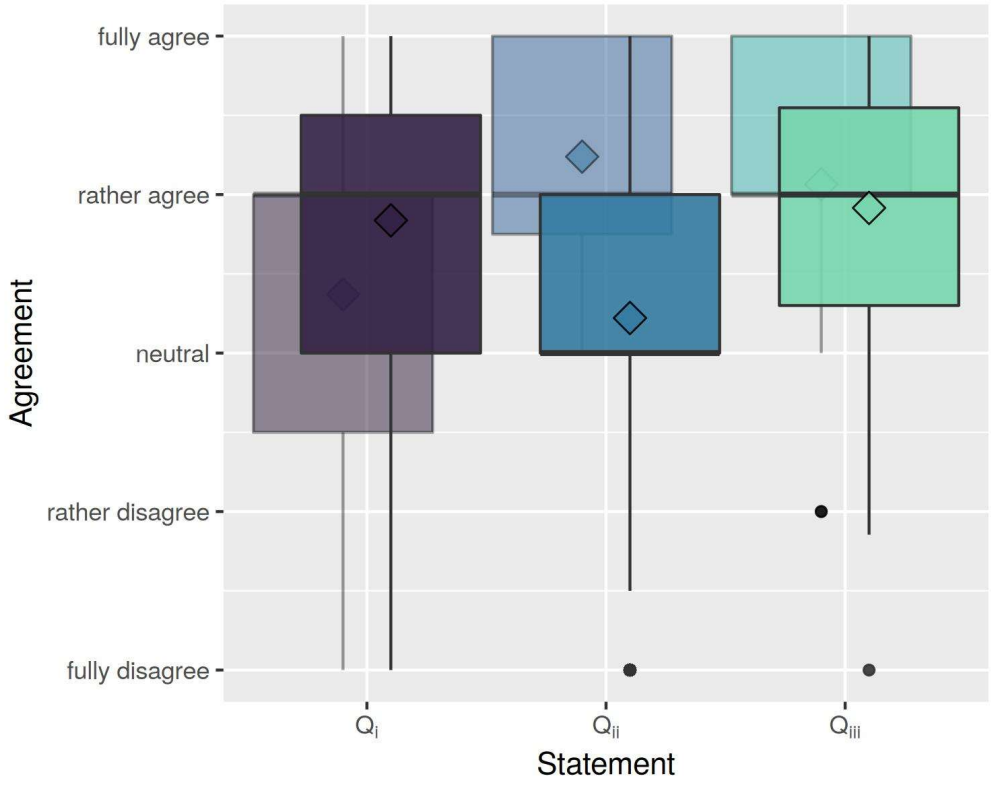

Figure 8. Boxplots of an agreement to the three questions. The faded, slightly left-shifted boxes are for the results of the former study [6], while the right-shifted boxes are SKQuest data.

3.3.2. Mapping Demographic Variables and Replication of Statements

Regarding the demographic variables, some variables map directly. For example, $isCustomer and $roleQA can map directly between both studies. While there are some differences in possible values, the budget maps quite easily to the original study's "D-Size"; however, there are two variables that have no direct counterpart in the SKQuest data:

The earlier survey distinguished between "Is Software" people and "others". The SKQuest data does not have an equivalent; although, it has variables $expSwDev and $expYDom that capture software development experience and domain experience. When software experience is bigger than domain experience, we assume it is a software person. Since the variables have different values, we normalize both to the range [0, 1].

The original study distinguished five roles and assigned a numeric value based on hierarchy level to each role: engineers (1), system engineers (2), project and PA managers (3), team leaders (4), and C-level managers (5). In the SKQuest data set, the roles slightly differ. We tried to map similar roles to similar values and assigned configuration managers and developers/engineers (1), engineering leads (2), quality/product assurance and project managers (3), heads of organizational units (4), and C-level managers (5). Students and

financial administrators are left out since they have no counterparts. If a respondent marked several roles, we used the mean value.

Table 7 places the original results next to the results from the attempt to replicate the results. For Qi (agreement with the statement about the emperor's new clothes), we do not find similar dependencies on demographic factors, although, there are two statistically significant results in the original study. More than that, the original inverse relationship for software people might, in fact, be positive. For Q_{ii}, we obtained two statistically significant correlations, where in the original there were none; albeit there were strong positive coefficients in the original study, too. For Q_{iii}, we have a comparable situation. While there is a statistically significant correlation in this column, it is for a different demographic factor.

Table 7. Comparison of Pearson correlation coefficients for original vs. replicated results. Statistically significant correlations with $p < 0.05$ are marked with *.

Demographic Factors	Q_i	Q_{ii}	Q_{iii}
Size	0.49 vs. 0.08	0.45 vs. 0.29 *	0.57 * vs. 0.11
Hierarchy level	0.31 vs. 0.08	0.15 vs. 0.03	−0.03 vs. 0.13
Is Customer?	0.43 * vs. 0.03	0.21 vs. 0.04	−0.07 vs. −0.03
Is product/quality assurance?	0.36 vs. 0.00	0.07 vs. 0.28 *	0.16 vs. −0.01
Is Software?	−0.49 * vs. 0.15	−0.01 vs. 0.03	0.28 vs. 0.18 *

Using the original and replicated coefficients as input variables to a Spearman correlation test, we obtained a statistically insignificant $r_s = 0.13$, i.e., the original results could not be replicated. However, we still find that demographic factors such as organization size, project budget, quality assurance view, and being a software person positively correlated with seeing issues of transparency in software development.

4. Discussion

The goal of this paper is to provide insights into the SKQuest data set and how it can be used so that other researchers can investigate their own questions. We demonstrated the data set's content and use by investigating four questions:

Question 1, i.e., whether transparency is a problem in software development projects, is confirmed. Transparency, or for that matter, missing transparency, is a strong influence factor for satisfaction with a project, its processes, and the resulting product. It is expected to make project execution, processes, and the product much better. The risk that software projects may fail, and that this becomes visible only late, was a latent fear of many respondents.

Question 2 deals with the desire for more transparency in software projects. First, we noted that participants were willing to accept more transparency. This must be seen in the context that respondents feel mostly well informed about their respective software development projects and also see that transparency can have undesired effects; however, desired effects are more probable to occur.

Question 3 investigates whether metrics can improve issues with transparency. Our respondents seemed to agree since they were willing to invest up to 10% of project money in metrication. Yet metrics play only a minor role among different activities that serve to generate transparency; the most used and successful among them being intermediate software releases and documentation. However, metrics and frequent delivery are deemed valuable by suppliers, internal developers, and especially customers. The individual metrics described by ECSS are mostly seen as beneficial, although there is a big variance.

Lastly, Question 4 looks at how metrication can be improved and improve the current situation. There seemed to be two opposing philosophies to either report metrics as rapidly as possible, even in real time, or to report them only very sparingly. Internal developers had a slight tendency toward more frequent reporting, whereas suppliers might prefer

more seldom reporting. Measurements for functionality, reliability, and maintainability were most desired by the respondents.

As a second means of demonstrating the use of the SKQuest data set, we tried to replicate the results of a more limited, earlier study. It was possible to address the problem with the SKQuest data. The basic results regarding the fundamental attitudes were similar. Just as in the former study, there was an agreement with the emperor's-new-clothes statement, the desire for more transparency is weaker but still measurable, and metrics are helpful.

However, findings regarding the concrete effects of specific demographic factors could not be confirmed. This could potentially speak against the usefulness of the SKQuest data set, yet the reasons are not necessarily problems of the SKQuest data or the necessary mapping but might also be due to the small size of the original study.

In fact, it might be a strength of the SKQuest data set that it allows checking assumptions in existing research works and standards. For example, we quantified the perceived importance of processes for product quality with 0.76 on a range from 0 to 1. This perceived importance is lower in agile development. Thinking of the agile manifesto—which proclaims "Individuals and interactions over processes and tools"—this is not very surprising. Yet, with 0.73 for partially or fully agile projects compared to 0.81 for partially or fully traditional projects, the difference is small.

4.1. Potential Effects on Existing and Future Standards

The main contribution of this paper is the SKQuest data set. The data can be used by practitioners and researchers as an empirical basis for their research. The analyses carried out in this paper demonstrate only a part of the variables contained in the data set. Furthermore, there are many more ways in which the variables can be combined to investigate new questions.

For example, measurement standards such as ECSS-Q-HB-80-04A [9] contain a list of stakeholders they target. One could test for a correlation between a respondent's role and how he perceives the usefulness of individual metrics. Of course, one would expect high correlations if the standard were well grounded. Negative correlations might be an argument to revise the standard (and, while further analysis is needed, this may be the case here).

The data set is also a research tool that enables the systematic design of software metrics, e.g., by showing in what areas practitioners see a need for new metrics. The data contains a manifold of information on how metrics can be used to make projects more successful. The data also enables empirically grounded research on how metrics can improve the efficiency of the software lifecycle, not limited to, but particularly with respect to transparency.

The results presented here are a plea to pay more attention to transparency in customer–supplier relationships in software development and the standards that address this business relationship. However, the paper may also serve as a starting point for future research avenues in academia and industry.

4.2. Threats to Validity

The main threat to the validity of the SKQuest data set is that the questionnaire was exceedingly long for an online survey. This has several implications:

The number of responses. We attracted quite a few visitors to the questionnaire but only a small fraction started the survey. One participant noted at the end: "Survey is very interesting but a little bit too long, don't be surprised if you don't get that many answers".

Survey completion rate. The survey completion rate was about 4% across all participants. Participants attracted through advertising had only a 2.7% chance of completing, while participants addressed personally completed in 41% of the cases.

Participant endurance. One participant remarked: "I think the durvey [sic!] is very extensive. No answer to all questions". Answers to some question items were missing. Moreover, some participants did not take the time to seriously answer every question.

While participants could skip most questions by just proceeding to the next page (and we explicitly pointed that out), some participants still seem to have given simple responses, e.g., fully agree to all items in a set. While we tried to identify and flag respective responses as bogus, this might not have worked out in all cases.

A major problem of the replication part is that the variables of both studies do not match exactly. Instead, we had to create a mapping between variables, which is a relevant source of insecurity.

5. Conclusions

This paper presents the SKQuest data set. With its 190 variables across 114 valid and complete responses, the data set opens many possibilities for future research on interesting and challenging research avenues in both academia and industry. The data set helps to understand metrics as a means of improving the efficiency of the software lifecycle. It adds value as a tool to systematically design software quality metrics during the software development lifecycle.

As a demonstration of what can be done with the data set, we analyzed the data to answer four broad questions from a bouquet of different angles:

- Transparency is a problem in software development projects, i.e., stakeholders—in particular, customers—miss transparency. Increased transparency leads to higher satisfaction with project execution, improves processes, and reduces the fear of overlooking project risks.
- Although respondents feel quite well informed and acknowledge certain risks from transparency itself, there is a desire for more transparency.
- Metrics are not the primary means of improving transparency in a software development project, but they can help.
- To improve the benefit of software metrics, they should be exchanged between supplier and customer. However, opinions regarding delivery frequency are polarized. Metrics for functionality, reliability, and maintainability are desired the most.

As a second demonstration, we tried to replicate results from an earlier and more limited study. The basic questions of whether there is truth in the emperor's-new-clothes statement, whether more transparency is desired, and whether ECSS metrics can help, could be affirmed. However, the specific demographic influence factors that the original study found could not be confirmed. While we cannot preclude that the reason may be the necessary mapping, it might also be artifacts in the much smaller original study.

The demonstrations show that metrics have their place in modern software development although there are limitations. The need for transparency has been high in the past and has not diminished. Metrics can help to mitigate these problems. Existing and future standards need to take transparency into account and consider metrics as a means of improving transparency. However, these results are only a demonstration, not the primary contribution of this paper.

Instead, we conclude that data from the SKQuest data set can be used to validate assumptions about software measurement and put research on an empirical basis. The data set may also prove valuable to empirically examine assumptions made by experts who wrote the existing standards.

Supplementary Materials: The following supporting information can be downloaded at: https://www.mdpi.com/article/10.3390/standards3020012/s1, The SKQuest data set in CSV format is provided as supplemental material to this paper. Permission is granted to use the data for research purposes given that an appropriate reference to the accepted version of this paper is made.

Author Contributions: Conceptualization, C.R.P. and R.G.; methodology, C.R.P. and R.G.; validation, R.G.; resources, C.R.P.; data curation, R.G. and C.R.P.; writing—original draft preparation, C.R.P.; writing—review and editing, C.R.P. and R.G.; visualization, C.R.P.; supervision, C.R.P.; project administration, R.G.; funding acquisition, C.R.P. All authors have read and agreed to the published version of the manuscript.

Funding: The SKQuest survey was contracted to Dr. Rainer Gerlich System and Software Engineering by OHB System AG on behalf of the German Space Agency at DLR in the frame of the AENEAS contract (number 50PS1602) on behalf of the German Federal Ministry for Economic Affairs and Climate Action.

Data Availability Statement: The SKQuest data set will be made available as supplemental material (see Supplementary Materials above).

Acknowledgments: Our biggest thank you goes to all of the participants of the survey who in total spent about 85 h of their lifetime on our survey, and additionally to those who we potentially misclassified as invalid responses. We hope that by publishing the data set, we can honor your contribution. We also thank our colleagues who kindly announced the survey to their contacts, and Alfred Hönle, who made room for this research in the scope of the AENEAS project. Last but not least, we thank Regina Gerlich and Thilo Nemitz for their advice on the survey and call-to-action design.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

This appendix provides, in Table A1, details on the different variables of the SKQuest data set. The question type (QT) describes the kind of question and how the data are presented in the data set.

- 5LI: 5-point Likert-scaled item. The range is usually from fully disagree (−2) over neutral (0) to fully agree (2).
- MLI: multiple 5-point Likert-scaled items.
- MC: multiple-choice questions, i.e., multiple-answer options can be selected. The responses are stored as multiple boolean yes/no values for the respective options.
- NT: number entered as text.
- SC: single choice question; only one answer option can be selected.
- YN: single choice question with only yes or no answers.

Some questions were only presented to certain subgroups of participants, i.e., customers, suppliers, and for internal development. For example, one might not expect to receive response data from a customer to a "suppliers only" question. However, it should be noted that respondents can be part of several groups, i.e., it is still possible that results are present for respondents belonging to other groups if they were also in the target group.

Table A1. A detailed structure of the SKQuest data set including the survey part, question type (QT), and individual question items with answer options. $key denotes the data column used in the data set to store the responses.

Topic	QT	Question Item and Answer Options
Introduction	n/a	Background and organization of the survey (e.g., context, topic, duration, etc.), information regarding anonymity, and voluntariness of participation.
Metadata	n/a	$mDuration: Duration of the interview in seconds. $mComplete: True iff the interview was completed successfully. $mCompleteness: Estimate how complete the response is, i.e., to how many logical blocks of questions there is an answer. Note that complete responses may have $mCompleteness < 1 since respondents might have skipped questions. $mBogus: True iff a participant is suspected of not having answered seriously, as determined by manual analysis. We recommend filtering these out; however, since we may have misjudged, we leave the responses in the data set.
		Demographics
Involved in software	YN	$involvedInSw: Whether the participant is involved in at least one project that includes the development, procurement, contracting, or integration of software.

Table A1. *Cont.*

Topic	QT	Question Item and Answer Options
Domain	MC	Domains the participant works in. While this item provided over 20 answer options, the values are mapped to only two values for anonymization reasons: $domAerospace: Whether the participant is working on aerospace projects. $domNonAero: Whether the participant is working on non-aerospace projects.
Office location	~~SC~~	Country where the participant's office is located. Data is not included for anonymization reasons.
Company size	~~SC~~	Size of the participant's company/institution in five predefined size levels. Data is not included for anonymization reasons.
Experience	MC	Several multiple-choice questions regarding experience: $expYDom: Years of professional experience in the domain: "<5", "5–10" (precisely: more than 5 but no more than 10 years), "10–20" (precisely: more than 10 but no more than 20 years), ">20". $expSwDev: Software development experience: "Very low" to "very high". $expSwMet: Software metrics experience: "Very low" to "very high".
Development culture	SC	Background of the participant to give a rough indication of the kind and culture of development work. $culture: "Is your work more focused on research or products?" Values: Fundamental (4) or applied research (3) vs. product development (2) or manufacturing (1). $pubOrPriv: The variable stores information on whether the respondent's organization is a "private" (for-profit) company or a "public" institute/institution.
		Project demographics
Choose "the project"	n/a	Upon reaching this part and before continuing with the questionnaire, participants were instructed to now think of one concrete current or past project. For the remainder of this survey, participants should answer project-related questions with respect to this project.
Project role	MC	The role or roles that the participant fulfilled in the context of the project. $roleClvl: C-level management. $roleHead: head of division/department/team/organizational unit. $rolePM: project manager. $roleEngL: head of engineering/development. $roleCM: configuration manager. $roleDev: engineer or developer. $roleQA: (software) quality or product assurance. $roleAdm: controller, project administrator, legal support. $roleStud: student; originally not included but frequently named as "other" category. $roleOth: other.
The agility of project management	SC	$agility: How is the project managed? Fully traditional, Rather traditional, Equally agile and traditional, Rather agile, and Fully Agile.
Project budget	SC	$budget: Average overall annual budget of the project. Values: "<100 K €", "100 K–1 M €", "1 M–10 M €", and ">10 M €".
Customer vs. supplier	MC	Customer and supplier roles in projects are often very distinct views. This aspect needs to be reflected in the questions presented to participants. The participants' responses to this question, therefore, had an enormous impact on what the rest of the survey looked like. "Software is developed … " $isSupplier: " … for an external customer". $isCustomer: " … by an external supplier". $isDev4Self: " … by my own organization for our own purposes or our own products". Note: Some participants reported having problems classifying one of the three options. We, therefore, added an "Other" option, enabling all question items. However, this did not add usable results.
Public customer	SC	$custPP: "Is your direct customer in "the project" a public sector entity or a private entity?" (Only suppliers). Values: public, or private.

Table A1. *Cont.*

Topic	QT	Question Item and Answer Options
Status quo of project execution, product quality, and transparency		
Satisfaction with project/product quality	MLI	Status quo of and satisfaction with project execution and product quality. "I am completely satisfied with … " $ssSoftware: " … the software delivered by our organization". (Only suppliers). $ssProcess: " … the compliance to the processes used for software development for our customer". (Only suppliers). $ssEfficiency: " … the efficiency of the processes used for software development for our customer". (Only suppliers). $scSoftware: " … the software delivered to me by my external software supplier". (Only customers). $scProcess: " … the compliance to the processes used by my external software supplier". (Only customers). $scEfficiency: " … the efficiency of the processes used by my external software supplier". (Only customers). $soSoftware: " … the software produced by my own organization for our own purposes". (Only internal development). $soProcess: " … the compliance to the processes used for the development of software by my own organization for our own purposes". (Only internal development). $soEfficiency: " … the efficiency of the processes used for the development of software by my own organization for our own purposes". (Only internal development).
Satisfaction with transparency	MLI	Satisfaction with the status quo of transparency in the project. $ssInfoStatus: "I feel well informed about the development status quo of the software delivered by us to our external customer in the project". (Only suppliers). $ssSurprises: "Surprises (e.g., regarding schedule or cost) happen regarding the development of software delivered by us to our external customer in the project". (Only suppliers). $ssInfoQuality: "I feel well informed about the quality (e.g., functionality, dependability, …) of software developed and delivered by us to our external customer in the project". (Only suppliers). $ssOverinfo: "The external customer knows too much about the project and our processes". (Only suppliers). $scInfoStatus: "I feel well informed about the development status quo of the software delivered to us by our external suppliers in the project". (Only customers). $scSurprises: "Surprises (e.g., regarding schedule or cost) happen regarding the development of software delivered to us by our external customer in the project". (Only customers). $scInfoQuality: "I feel well informed about the quality (e.g., functionality, dependability, …) of software developed and delivered to us by our external customer in the project". (Only customers). $scOverinfo: "Receiving less information from our external supplier(s) in the project would be fine for me". (Only customers). $soInfoStatus: "I feel well informed about the development status quo of the software developed internally in the project". (Only internal development). $soSurprises: "Surprises (e.g., regarding schedule or cost) happen regarding the development of software developed internally in the project". (Only internal dev). $soInfoQuality: "I feel well informed about the quality (e.g., functionality, dependability, …) of software developed internally in the project". (Only internal dev). $soOverinfo: "Other organizational entities of my organization get to know too much about the project".

Table A1. *Cont.*

Topic	QT	Question Item and Answer Options
Transparency activities	MC	Activities that are in place that increase transparency. "Which of the following activities are already part of the development process of the software?" $tRegRel: "Regular releases of intermediate software versions to the customer". $tRegManMeet: "Regular face-to-face meetings between managing representatives of the different stakeholders". $tRegTechMeet: "Regular face-to-face meetings between the technical staff of the different stakeholders". $tRegReflect: "Regular reflection of the effectiveness of the process and possible improvement". $tFocusInteract: "Focus on interaction between individuals rather than between institutions". $tUseMet: "Regular use of software metrics by the software team". $tDlvrMet: "Regular delivery of software metrics to the customer". $tCommMet: "Regular communication of software metrics to other stakeholders within the developing organization". $tMiles: "Regular milestone or phase reviews with quality gates". $tDoc: "Detailed documentation". $tDocUpd: "Regular updates of documentation". $tTeamVisit: "Regular visits to the software team by the customer".
Emperor's New Clothes	5LI	Agreement with the anecdote of the metaphorical comparison between software developers and the weavers of the emperor's new clothes.
Metric use	SC	Current use of metrics in the project. $tsMetReport: "How many metrics do you report regularly for your project?" (Only suppliers or internal development). $tcMetReport: "How many metrics are reported regularly to you for your project?" (Only customers or internal development). Values: "None", "<5" (Up to 5), "5–10" (More than 5 but no more than 10), and ">10" (More than 10).
Metric format	SC	Current use of machine-readable data formats for reporting metrics. $tsMetMachRead: "Do you report the software metrics in a machine-readable format (e.g., XML and CSV)?" (Only suppliers or for internal development). $tcMetMachRead: "Are software metrics reported in a machine-readable format to you (e.g., XML and CSV)?" (Only customers or for internal development). Values: Yes, Partially, and No.
		The role of transparency for project success
Process and quality	SC	This item establishes a relationship between process quality and product quality, i.e., how important the process is deemed for product quality. $relStrProcQual: "How important do you consider the software development process for the quality of the developed product?" Note: The question item was implemented as a slider. Values: 0.0 (unimportant), 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, and 1.0 (important).
		A broad analysis of the wide range of effects that transparency can have on the project, its execution, and its environment. "Increased transparency … " $tEffMeetObj: " … reduces the risk of not meeting the development objectives". $tEffMeetBudget: " … reduces the risk of exceeding the budget". $tEffMeetQual: " … reduces the risk of insufficient product quality". $tEffIncEffort: " … substantially increases the effort on the side of the information provider".

Table A1. *Cont.*

Topic	QT	Question Item and Answer Options
Effects of transparency	MLI	$tEffNoBenefitS: " … has no benefits for the information provider". $tEffNoBenefit: " … benefits no one". $tEffImpStakCom: " … improves communication between stakeholders". $tEffImpTrustInS: " … strengthens the customer's trust in the external supplier(s)". $tEffExploitS: " … gives the customer leverage to take advantage of the external supplier(s)". $tEffInfoLeak: " … increases the risk of information leaks to competitors". $tEffImpOwnrspC: " … leads to a stronger sense of product ownership by the customer". $tEffImpCtrl: " … improves control over the development progress, reducing schedule risks". $tEffIncDiscuss: " … also implies more effort for discussing the information with the customer". $tEffProcImpInsight: " … gives better insight into possible improvements of development processes". $tEffIncCResponsibility: " … increases the customer's co-responsibility for failure". $tEffAssessQuality: " … allows for more objective assessment of product quality". $tEffImpCommitC: " … leads to improved commitment by the customer". $tEffAssessProcImp: " … allows for more objective assessment of process improvements". $tEffAssessEmpPerf: " … enables more objective performance assessment of team members and employees". $tEffInappEmpMonitor: " … makes employees feel inappropriately monitored". $tEffInappSMonitor: " … may be considered inappropriate monitoring by external supplier(s)". $tEffExploitEmp: " … enables employers to take advantage of their employees". $tEffEmpReplaceable: " … makes people become more easily replaceable".
Transparency and project success	SC	Expectations of customers and suppliers of how increased transparency affects the quality of process, product, and project execution, i.e., whether it gets worse or better. $tEffSProc: "Regarding the software provided by you, if transparency increases relative to its current state, the process … " (Only supplier). $tEffSProd: "Regarding the software provided by you, if transparency increases relative to its current state, the product … " (Only supplier). $tEffSProj: "Regarding the software provided by you, if transparency increases relative to its current state, the project execution … " (Only supplier). $tEffCProc: "Regarding the software provided to you, if transparency increases relative to its current state, the process … " (Only customer). $tEffCProd: "Regarding the software provided to you, if transparency increases relative to its current state, the product … " (Only customer). $tEffCProj: "Regarding the software provided to you, if transparency increases relative to its current state, the project execution … " (Only customer). $tEffOProc: "Regarding the software provided internally, if transparency increases relative to its current state, the process … " (Only internal development). $tEffOProd: "Regarding the software provided internally, if transparency increases relative to its current state, the product … " (Only internal development). $tEffOProj: "Regarding the software developed internally, if transparency increases relative to its current state, the project execution … " (Only internal development). Values: "(gets) much better" (5), "(gets) slightly better" (4), "(is) not affected" (3), "(gets) slightly worse" (2), and "(gets) much worse" (1).
		Increasing transparency
Acceptance of increased transparency	5LI	Likert-scale item measuring whether suppliers and development teams would accept an increase in transparency. $accIncT: "I would be willing to increase the transparency of our process and the current state of software development in the project for our customer(s)". (Only suppliers and internal development).

Table A1. *Cont.*

Topic	QT	Question Item and Answer Options
Activities that increase transparency	MC	Assessment of whether certain activities are useful for increasing transparency. Note: The role of metrics for increasing transparency is discussed separately and in more detail in the next block. "What other measures do you consider useful for increasing transparency?" $tIncARegManMeet: "More face-to-face meetings between managing representatives of the different stakeholders". $tIncARegTechMeet: "More face-to-face meetings between the technical staff of the different stakeholders". $tIncARegReflectCS: "More frequent reflection on the effectiveness of the process and possible improvements by customer and supplier". $tIncARegReflectTeam: "More frequent reflection on the effectiveness of the process and possible improvements by the software team". $tIncARegReflectOrg: "More frequent reflection on the effectiveness of the process and possible improvements by different units of the organization". $tIncACloserCoopCS: "Closer cooperation between customer and supplier". $tIncAFocusInteract: "Increased focus on the interaction between individuals rather than between institutions". $tIncAMilestones: "More milestone or phase reviews with quality gates". $tIncATeamVisits: "More frequent visits to the software team by the customer". $tIncADoc: "Provision of more detailed documentation". $tIncADocUpd: "Provision of more up-to-date documentation".
Increasing transparency through metrics	5LI	Several Likert-scale items measuring the respondents' assessment of whether metrics can increase transparency: $tIncAMetCFreq: "Transparency increases for the customer when software metrics are delivered to the customer more often". $tIncAMetCMore: "Transparency increases for the customer when more software metrics are delivered to the customer". $tIncAMetSFreq: "Transparency increases for the software team when metrics are used by the team more often". $tIncAMetSMore: "Transparency increases for the software team when more metrics are used by the team". $tIncAMetOFreq: "Transparency increases inside the organization when metrics are communicated to other stakeholders within the developing organization more often". $tIncAMetOMore: "Transparency increases inside the organization when more metrics are communicated to other stakeholders within the developing organization".
The usefulness of and increasing transparency with metrics		
Acceptable cost of metrics	NT	A percentage value of the yearly project budget that would be an acceptable effort for collecting metrics. $accMetCost: "What percentage of the yearly budget would you consider acceptable for regularly gathering software metrics in the project?" Values: An integer from 0 to 100.
Metric delivery frequency	SC	Recommended frequency of updating and exchanging measured software metrics. $metRecFreqS: "What would be a good frequency for delivering up-to-date software metrics in your project to your external customer?" (Only supplier). $metRecFreqC: "What would be a good frequency for receiving software metrics from your external software suppliers?" (Only customer). $metRecFreqO: "What would be a good frequency for updating software metrics for your internal software development?" (Only internal development). Values: "quarterly" (Up to once every three months), "monthly" (More often than once per quarter but not more often than once per month), "biweekly" (More often than once per month but not more often than once every two weeks), "weekly" (More often than once every two weeks but not more often than once per week), "daily" (More often than once per week), and in "real time".

Table A1. *Cont.*

Topic	QT	Question Item and Answer Options
Metric usefulness	MLI	Participants were asked to rate the overall usefulness of various software metrics. Only ten metrics out of the full set of 41 ECSS metrics were presented to participants. The metrics were selected randomly for each participant. $metric[…]: "[Short explanation of metric.] [Metric] is a useful software metric".
New metrics	MC	Upon what kind of metrics should research and development of new metrics focus? It was possible to choose individual product quality characteristics from the ISO-25000 and process effectiveness and efficiency. $newMetFunc: product quality, functionality. $newMetRel: product quality, reliability. $newMetMaint: product quality, maintainability. $newMetReuse: product quality, reusability. $newMetSfty: product quality, suitability for safety. $newMetSec: product quality, security. $newMetUsa: product quality, usability. $newMetEffi: product quality, efficiency. $newMetComp: product quality, compatibility. $newMetPort: product quality, portability. $newMetDevEffe: software process, development effectiveness. $newMetDevEffi: software process, development efficiency.
		Quality of tools and support for metrics
Open metrics database benefits	MLI	Estimated usefulness of a cross-project and cross-institutional database with project information and software metrics for different communities. "A cross-project and cross-institutional database of project information including software metrics would be useful to … " $metDbBenefitSpace: " … the space community". $metDbBenefitIndustry: " … industry". $metDbBenefitScience: " … science". $metDbBenefitDevCom: " … the software development community".
Open metrics database contribution	SC	Willingness to contribute data to a cross-project and cross-institutional database with software metrics, and whether such contribution would need to guarantee anonymity. $metDbContrib: "Would you personally be willing to provide such data to such a database?" Values: "openly" (Yes, even if the data are not anonymized), "anonymously" (Yes, if the data are anonymized), or "no".
Current metrics tool landscape	MLI	Statements regarding the availability and suitability of software metrics tools. $mtMissRelMet: "Current metrication tools do not support the metrics relevant to me". $mtPricy: "All metrication tools relevant to me are too expensive". $mtEffortAnalysis: "Assessing the metric data with current tools requires just too much effort". $mtOldData: "With current tools, metric data are always too old when it becomes available". $mtLackSources: "None of the tools available supports all the data sources I need". $mtDisjunct: "I need multiple tools, but they cannot be integrated into a complete solution". $mtNeedFormat: "We urgently need some standards for the exchange of metric data between tools". $mtNewSources: "I often have new data sources which need to be integrated with the metric collection tools". $mtProcMismatch: "Metrication tools are difficult to integrate with our processes". $mtITMismatch: "Available metrication tools don't fit with our IT". $mtApproveData: "There must be a process to approve metrics data before it is disclosed to the customer". $mtNeedOSS: "I would only use a metrication tool, if it is available under an open-source license".

Appendix B

Table A2 lists all metric variables in the SKQuest data set along with their metrics' chapter code in ECSS-Q-HB-80-04, its official name [and potential variant], and the brief description that was provided to participants during the survey.

Table A2. Short descriptions of 42 software metrics as presented to survey participants.

Code/Var. Name	Metric Name	Short Description
A.3.3.01 metricReqAlloc	Requirement allocation [var. swdc]	Percentage of Software Requirements allocated to Software Design Components.
A.3.3.01 metricSysReqAlloc	Requirement allocation [var. Sys2Sw]	Percentage of System Requirements allocated to Software Level Requirements.
A.3.3.02 metricReqImpl	Requirement implementation coverage	Percentage of correctly implemented and validated Software Requirements.
A.3.3.03 metricReqWithTBD	Requirement completeness	Percentage of Software Requirements containing TBC/TBD to be confirmed/defined.
A.3.3.04 metricVVCmplness	V&V coverage	Percentage of Software Requirements covered by at least one verification or validation activity.
A.3.3.05 metricBugHistory	SPR/NCR trend analysis	Evolution of open vs. closed bug reports over time.
A.3.3.06 metricReqClarity	Requirement clarity	Percentage of software requirements containing ambiguous phrases.
A.3.3.07 metricCheckDocSui	Suitability of development documentation	Percentage of positively answered questions in the checklist for documentation suitability.
A.3.3.08 metricCodingRuleCompl	Adherence to coding standards	Percentage of positively evaluated coding standard checks.
A.3.3.09 metricCPUsed	CPU margin	Minimum unused CPU capacity during operation.
A.3.3.10 metricMemUsed	Memory margin	Percentage of available memory used.
A.3.3.11 metricMcCabe	Cyclomatic complexity (VG)	Average cyclomatic complexity per source code module.
A.3.3.12 metricNesting	Nesting level	Maximum nesting level per source code module.
A.3.3.13 metricLocPerMod	Lines of code	Number of lines of code per module without comments/blank lines.
A.3.3.14 metricCommentLines	Comment frequency	Percentage of comment lines in source code per module.
A.3.3.15 metricReqTested	Requirement testability	Percentage of Software Requirements validated by a test.
A.3.3.16 metricFanOut	Modular span of control	The number of subroutines the functions call on average.
A.3.3.17 metricCoupling	Modular coupling	The median level of coupling between pairs of modules.
A.3.3.18 metricCohesion	Modular cohesion	The maximum level of cohesion of source code modules.
A.3.3.19 metricCheckRelAct	Process reliability adequacy	Percentage of positively answered questions in the checklist for performed reliability activities.

Table A2. *Cont.*

Code/Var. Name	Metric Name	Short Description
A.3.3.20 metricBranchCov	Structural coverage [var. branch]	Percentage of branches executed during testing.
A.3.3.20 metricMCDCov	Structural coverage [var. MC/DC]	Percentage of MC/DC coverage achieved during testing.
A.3.3.20 metricStatementCov	Structural coverage [var. statement]	Percentage of source code statements executed during testing.
A.3.3.21 metricOpenBugs	SPR/NCR status	Number of open bugs by criticality class over time.
A.3.3.22 metricCompBehavSw	Environmental software independence	Percentage of design components expected to maintain their correct behavior in a different software environment.
A.3.3.23 metricCompBehavHw	System hardware independence	Percentage of design components expected to maintain their correct behavior in a different hardware environment.
A.3.3.24 metricReuseChkl	Reusability checklist	Percentage of positively answered questions in the checklist for reuse potential.
A.3.3.25 metricModLineReuse	Reuse modification rate	Percentage of modified/added lines of existing/ reused software.
A.3.3.26 metricSafetyChkl	Safety activities adequacy	Percentage of positively answered questions in the checklist for performed safety activities.
A.3.3.27 metricSecurityChkl	Security checklist	Percentage of positively answered questions in the checklist for performed security activities.
A.3.3.28 metricAmbigUM	User documentation clarity	Percentage of sentences in the user manual containing ambiguous phrases.
A.3.3.29 metricUserDocCmpl	User documentation completeness	Percentage of sections within the user manual containing TBC/TBD to be confirmed/defined.
A.3.3.30 metricUserManSui	User manual suitability	Percentage of positively answered questions in the checklist for user manual suitability.
A.3.3.31 metricCheckMMI	Adherence to MMI standards	Percentage of positively answered questions in the checklist for adherence to man-machine-interface standards.
A.3.3.32 metricCmmiLvl	Process assessment [ECSS-HB-Q-80-02]	The assessed level of the contractor's process capability and maturity.
A.3.3.33 metricMilestoneMet	Milestone tracking	Difference between planned and actually achieved dates for project milestones.
A.3.3.34 metricEffortMet	Effort tracking	Estimated and actual effort figures for each ongoing or completed task relevant to the software project.
A.3.3.35 metricSizeStable	Code size stability	Estimated and actual physical lines of code for each major design component.
A.3.3.36 metricReqStable	Requirement stability	Percentage of software requirements added modified and deleted since the last software version.
A.3.3.37 metricOpenItems	RID/action status	Number of open action items from milestone reviews over time.
A.3.3.38 metricVVProgress	V&V progress	Percentage of successfully completed verification and validation activities.

References

1. Guanter, L.; Kaufmann, H.; Segl, K.; Foerster, S.; Rogass, C.; Chabrillat, S.; Kuester, T.; Hollstein, A.; Rossner, G.; Chlebek, C.; et al. The EnMAP Spaceborne Imaging Spectroscopy Mission for Earth Observation. *Remote Sens.* **2015**, *7*, 8830–8857. [CrossRef]
2. Prause, C.R.; Bibus, M.; Dietrich, C.; Jobi, W. Software product assurance at the German space agency. *J. Softw. Evol. Proc.* **2016**, *28*, 744–761. [CrossRef]
3. Donaldson, S.E.; Siegel, S.G. *Successful Software Development*, 2nd ed.; Prentice Hall PTR: Upper Saddle River, NJ, USA, 2001.
4. Tu, Y.-C.; Tempero, E.; Thomborson, C. Evaluating Presentation of Requirements Documents: Results of an Experiment. In *Requirements Engineering*; Junqueira Barbosa, S.D., Chen, P., Cuzzocrea, A., Du, X., Filipe, J., Kara, O., Kotenko, I., Sivalingam, K.M., Ślęzak, D., Washio, T., et al., Eds.; Springer: Berlin/Heidelberg, Germany, 2014; pp. 120–134, ISBN 978-3-662-43609-7.
5. Boehm, B.W. *Software and Its Impact: A Quantitative Assessment*; RAND Corporation: Santa Monica, CA, USA, 1972.
6. Prause, C.R.; Hönle, A. Emperor's New Clothes: Transparency Through Metrication in Customer-Supplier Relationships. In *Product-Focused Software Process Improvement*; Kuhrmann, M., Schneider, K., Pfahl, D., Amasaki, S., Ciolkowski, M., Hebig, R., Tell, P., Klünder, J., Küpper, S., Eds.; Springer International Publishing: Cham, Switzerland, 2018; pp. 288–296, ISBN 978-3-030-03672-0.
7. Betta, J.; Boronina, L. Transparency in Project Management—From Traditional to Agile. *Adv. Econ. Bus. Manag. Res.* **2018**, *56*, 446–449. [CrossRef]
8. *ECSS-Q-ST-80C*; Space Product Assurance: Software Product Assurance. ECSS Executive Secretariat: Noordwijk, The Netherlands, 2017.
9. *ECSS-Q-HB-80-04A*; Space Product Assurance: Software Metrication Programme Definition and Implementation. ECSS Executive Secretariat: Noordwijk, The Netherlands, 2011.
10. *ISO/IEC/IEEE 15939:2017*; Systems and Software Engineering—Measurement Process. ISO: Geneva, Switzerland, 2017.
11. LamaPoll. Sichere Online Umfrage. Available online: http://www.lamapoll.de (accessed on 23 March 2023).
12. *ISO/IEC 25010:2011*; Systems and Software Engineering—Systems and Software Quality Requirements and Evaluation (SQuaRE)—System and Software Quality Models. ISO: Geneva, Switzerland, 2011.
13. Basili, V.R.; McGarry, F.E.; Pajerski, R.; Zelkowitz, M.V. Lessons learned from 25 years of process improvement. In Proceedings of the 24th International Conference on Software Engineering—ICSE '02, Orlando, FL, USA, 19–25 May 2002; Tracz, W., Magee, J., Young, M., Eds.; ACM Press: New York, NY, USA, 2002; p. 69, ISBN 158113472X.
14. Chidamber, S.R.; Kemerer, C.F. A metrics suite for object oriented design. *IEEE Trans. Softw. Eng.* **1994**, *20*, 476–493. [CrossRef]
15. Saraiva, J.; Barreiros, E.; Almeida, A.; Lima, F.; Alencar, A.; Lima, G.; Soares, S.; Castor, F. Aspect-oriented software maintenance metrics: A systematic mapping study. In Proceedings of the 16th International Conference on Evaluation & Assessment in Software Engineering (EASE 2012), Ciudad Real, Spain, 14–15 May 2012; IET: Hong Kong, China, 2012; pp. 253–262, ISBN 978-1-84919-541-6.
16. Bouwers, E.; van Deursen, A.; Visser, J. Towards a catalog format for software metrics. In Proceedings of the 5th International Workshop on Emerging Trends in Software Metrics, ICSE '14, 36th International Conference on Software Engineering, Hyderabad, India, 3 June 2014; Counsell, S., Marchesi, M.L., Visaggio, A., Zhang, H., Venkatasubramanyam, R., Eds.; ACM: New York, NY, USA, 2014; pp. 44–47, ISBN 9781450328548.
17. Sayyad Shirabad, J.; Menzies, T.J. The PROMISE Repository of Software Engineering Databases. 2005. Available online: http://promise.site.uottawa.ca/SERepository (accessed on 13 March 2023).
18. Vogel, M.; Knapik, P.; Cohrs, M.; Szyperrek, B.; Pueschel, W.; Etzel, H.; Fiebig, D.; Rausch, A.; Kuhrmann, M. Metrics in automotive software development: A systematic literature review. *J. Softw. Evol. Proc.* **2021**, *33*, e2296. [CrossRef]
19. Le Son, H.; Pritam, N.; Khari, M.; Kumar, R.; Phuong, P.; Thong, P. Empirical Study of Software Defect Prediction: A Systematic Mapping. *Symmetry* **2019**, *11*, 212. [CrossRef]
20. Choras, M.; Springer, T.; Kozik, R.; Lopez, L.; Martinez-Fernandez, S.; Ram, P.; Rodriguez, P.; Franch, X. Measuring and Improving Agile Processes in a Small-Size Software Development Company. *IEEE Access* **2020**, *8*, 78452–78466. [CrossRef]
21. Brüggemann, S.; Prause, C. Status quo agiler Software-Entwicklung in der europäischen institutionellen Raumfahrt. In Proceedings of the Deutscher Luft- und Raumfahrtkongress (DLRK), Friedrichshafen, Germany, 4–6 September 2018; pp. 1–8.
22. Ofem, P.; Isong, B.; Lugayizi, F. On the Concept of Transparency: A Systematic Literature Review. *IEEE Access* **2022**, *10*, 89887–89914. [CrossRef]
23. Saraiva, R.; Medeiros, A.; Perkusich, M.; Valadares, D.; Gorgonio, K.C.; Perkusich, A.; Almeida, H. A Bayesian Networks-Based Method to Analyze the Validity of the Data of Software Measurement Programs. *IEEE Access* **2020**, *8*, 198801–198821. [CrossRef]
24. de Vaus, D.A. *Surveys in Social Research*, 5th ed.; Routledge: London, UK, 2002; ISBN 0415268575.