Emperor's New Clothes: Transparency through Metrication in Customer-Supplier Relationships

Christian R. Prause[1] and Alfred Hönle[2]

[1] German Aerospace Center, christian.prause@dlr.de
[2] OHB System AG, alfred.hoenle@ohb.de

Abstract. Space projects, and development of software embedded in these systems, are complex, sometimes costing hundreds of millions of Euros and involving several tiers of suppliers. An important means of improving mutual understanding is to increase transparency of the development status between customers and suppliers. We raise the problem of transparency in complex projects to the reader's attention, and, relying on results of a small survey of practitioners, propose to use ECSS software metrics/KPIs as a mitigation. We present our metrication infrastructure, and describe issues to be considered when implementing an early metrication programme in a real-world, industry space project.

Keywords: software metrication · management · customer-supplier transparency · embedded software · aerospace · ECSS · KPI

1 Introduction

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.[3] The situation today might be not as dramatic as provocatively stated almost 50 years ago. But since the early days of the *software crisis* and *software engineering*, size and complexity of software have kept increasing. The ratio of costs for all hardware and software of 10:1 reversed to 1:2 in US missions. The embedded flight software alone accounts for up to 20% of satellite costs. It allows more challenging missions and to reply to increasingly demanding user requirements [1, 12, 15, 25]. Although a metrication standard exists, not much attention has yet been paid to it in practice. *RQ: Are ECSS*[4] *metrics a practicable way to improve transparency of software development in customer-supplier relationships of space missions?*

[3] Statement by an Air Force decision maker as reported by [6].
[4] The European Cooperation for Space Standardization (ECSS) is a cooperative effort of ESA, national space agencies, and industry to develop and maintain a single, coherent set of standards for hardware, software, and other activities ([12], cf. [22]).

Original source of publication: Prause, C.R., Hönle, A. (2018). Emperor's New Clothes: Transparency Through Metrication in Customer-Supplier Relationships. In: Kuhrmann, M., Schneider, K., Pfahl, D., Amasaki, S., Ciolkowski, M., Hebig, R., Tell, P., Klünder, J., Küpper, S. (eds.) Product-Focused Software Process Improvement. PROFES 2018. Lecture Notes in Computer Science, vol. 11271. Springer, Cham.

The final publication is available at Springer via https://doi.org/10.1007/978-3-030-03673-7_21

Self-archived at: https://www.drprause.de/publications/

- We describe our practitioners' view on transparency in the pervasive customer-supplier relationship of European space projects (Section 2),
- survey industry and agency practitioners whether they wish for improved transparency, and if they expect benefits from software metrics (Section 3),
- present the software architecture of the AENEAS metrication infrastructure and define a data protocol for delivering the data (Section 4), and
- identify early lessons from implementing and using metrication (Section 5).

Finally, we conclude (see Section 6).

2 Background: Transparency and Software Metrics

Many spacecraft are one-of-a-kind devices with uncommon and custom-built hardware and software. An identical unit is rarely built. If the mission goal is not reached, for whatever reason, there is no second chance. In these systems, software fulfills more and more critical functions. A single failure, in the worst case, can mean that the spacecraft and its mission are lost (cf. [22]). While most software can be updated in flight, fixing bugs in decades old software (without introducing new ones) is challenging. Preparing a mission, which includes making of the spacecraft and its ground support equipment, may take more than ten years. Its cost can be the hundreds of millions of Euros. What adds to project complexity is that there are many stakeholders like the final customer, prime contractor(s), several tiers of suppliers, scientific users, and potentially material providers on some other contractual basis. Space systems are similar to other large, one-of-a-kind things like airports or nuclear power plants; and likewise, as opposed to buying off-the-shelf, contract partners need visibility or *transparency*. Customers get involved in the development (process) so that problems can be identified early and reacted upon in a timely manner [8] (see also [22]). Donaldson and Siegel [8] note: "Successful software development is first and foremost an ongoing exercise in effective communication between the customer and the seller throughout a software project."

Transparency through customer involvement is a major motivator for agile development. However, it is not considered as a home ground for high-reliability products, and there are some issues with contract design and public offer bidding [7, 10, 14, 19, 18]. Successful ways to improve transparency are (i) *stages* with extensive[5] reviews at system and subsystem levels, which are a key element in ECSS standards, and (ii) *measurements* of software process and product [8]. Hence, for us, *software metrics* are not only typical source code metrics like cyclomatic complexity, but also other Key Performance Indicators (KPI).

Software engineering is "a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is the application of engineering to software" [16]. Unsurprisingly, metrication has a long history

[5] Stages are central to ECSS (cf. [11]). Yet, if conducted only half-heartedly, they cannot satisfy transparency needs and should then be considered a waste of time [8].

in software engineering. Ebert and Dumke [9] point out that the more mature an engineering discipline is, the more measurements are used, that one can only control what can be observed and measured, and that in the global market organizations cannot be successful without paying attention to their performance (including quality). Without metrication, different understandings of process goals and progress may become apparent late, leading to discussion, disappointment, rework, and/or contractual adjustments; e.g., regarding unit test coverage (cf. [24]). In software development, many of the relevant artifacts to be measured are already stored in information systems. Once interfaces are set up, metrication incurs practically no additional cost. Metrication reduces risks for suppliers by reducing room for misinterpretation, misunderstandings and speculation, and may promote timely dealing with certain customer expectations. The customer benefits from reduced risks for the business partners (e.g., promoting Deming's ideal of long-term partnership, lower risk surcharges, etc.), better adherence to plans, and confidence to get the desired project result (cf. [12]).

Metrication in space projects. In space projects, obtaining highly dependable and high quality software within time and budget is a priority objective. Measurements are the only way to quantitatively assess the quality of a process or product, to know what to do better, and what to look at to understand where lessons learned can support improvement. Metrication provides a management tool for keeping the project on track and achieve the intended product quality. It is used to *characterize* existing processes and product status, to *evaluate* project status to detect deviations and regain control, to *predict* by gaining an understanding of process-product relationships, and to *improve* product quality and process performance [12]. The main software standards in the ECSS are ECSS-E-ST-40C [11] and ECSS-Q-ST-80C [13]. They both lay the foundation for metrication by specifying respective metrication requirements.

ECSS-Q-HB-80-04A is based on ISO-15939 adapted to space software projects and describes metrication as part of the project [17, 12]. It covers topics like selecting metrics, metrication planning, interfaces with other processes, data collection aspects, feedback from metrics into process and product, continuous improvement through metrics. Most of the document is an appendix with 40+ process, product and object-oriented metrics listing details like the addressed quality characteristics, owner/producer, target audience, evaluation method, calculation formula, and in which phases and to software of what criticality the metrics are applicable. After defining a project's quality model, the metrics shall be tailored to the model, and to company culture and experience [12].

Related Work. Basili et al. [5] used metrics successfully to introduce improvements at NASA, and advance software engineering as a whole. An ecosystem of tools evolved alongside computer-assisted software measurement and evaluation (CAME) [9]. Current studies of tools often focus on code metrics, e.g., [26, 20], which are the basis of software visualization (e.g., [27]). Practitioners in applied research projects mostly use metrics like lines of code, code coverage, or cyclomatic complexity; however, over 50% of projects use no metrics at all [23].

Question	*Type*	*Question (abbreviated), response options*
D-Role	MC, FT	What are your roles? (Engineer, project manager, product/quality assurance, team/department head, ...)
D-Sw	SC	Is your main job in the area of software? (yes/no)
D-CuSu	MC	Do you act mainly as customer or supplier?
D-Size	FT	Estimated annual average total cost of projects where you fulfill your role(s)?
Q-i	5Ls	Do you agree to the statement about software and the emperor's new clothes?
Q-ii	5Ls	Do you wish for more transparency of software development?
Q-iii	5Ls	Would regular delivery of ECSS metrics (some examples given) help you fulfill your role?

Table 1: Questionnaire design. Responses: SC/MC=single/multiple choice, FT=free text, 5Ls=5-point Likert-scaled (1=fully disagree, ..., 5=fully agree).

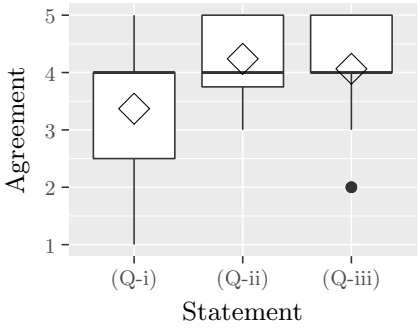
An earlier (unpublished) project to implement ECSS metrics used SQuORE (cf. [2]). Yet only few of its generic data providers fitted to space processes, and would have had to be developed anew for AENEAS. Data export, and deliveries between customer and supplier were not inherent features, and vendor lock-in impended. With respect to this, the open-source platform Alambic might become a suitable alternative [3].

3 Pre-Test Survey of Need for Metrics

In order to better understand the issue of transparency of software development in space systems development, we organized a small-scale survey based on a no-more-than-ten-minutes questionnaire. Table 1 shows the questionnaire design. We invited key personnel from industry and a space agency via email, providing a paper version for anonymous answering but participants could also respond via email. Participation was voluntary, with a response rate of $\approx 40\%$.

Survey participants (N=23) see themselves as customer (12) and supplier (13), as PA/QA (6), and as software people (13), they fulfill engineering (11) and more management-heavy roles (12). The average survey participant fulfills his roles in projects with an annual budget of 70M EUR. Figure 1a shows that average respondent (rather) approves all three questions. However, in particular the statement regarding software and the weavers of the emperor's new clothes is polarizing. The responses range from full agreement and to full disagreement, with both extremes not being outliers. Wishes for more transparency and perceptions of usefulness of metrics are clearly present. Figure 1b investigates the effects of personal attributes on opinions. Being involved in larger projects leads to more agreement (positive correlation) with all questions. Participants working in management-heavy roles[6], tend to see software development as more problem-

[6] We transformed D-Role responses to a $1-5$ score, e.g., engineers (1), system engineers (2), project and PA managers (3), team leaders (4) and C-level managers (5).



(a) Boxes go from lower to upper quartile. Diamond shows mean. Horizontal line marks median. Median of 3.5/4.5 results from responses "3.5"/"4.5" in an email.

Subj. attribute	*Q-i*	*Q-ii*	*Q-iii*
D-Size	0.49	0.45	0.57*
D-Role	0.31	0.15	-0.03
Is Customer	0.43*	0.21	-0.07
Is PA/QA	0.36	0.07	0.16
Is Software	-0.49*	-0.01	0.28

(*) = Statistical significance at $p < 0.05$. Note: Failed significance tests may be due to small N, and not necessarily mean there is no effect.

(b) Influence factors for agreement to questions Q-i, Q-ii and Q-iii as Pearson correlation coefficients r.

Fig. 1: Analysis of survey participants' responses to Q-i, Q-ii, and Q-iii. N=23.

atic but differences diminish regarding the wish for transparency and usefulness of metrics. Similar, but stronger, is this effect for customers as compared to suppliers; PA/QA personnel seem to value metrics more. Interestingly, for software people the effect is inverted: they rather reject Q-i, do not have transparency needs different from others, but might tend to see more value in metrics.

Limitations The survey was designed to be small-scale, i.e., we only had few question items, and invited few participants from two organizations only. Consequently, N=23 is small, with implications on statistical significance. Also, respondents with management- and software-heavy roles are not evenly distributed across customer and supplier roles, leading to bias in influence factor results.

4 AENEAS Project: Implementation and Field-Trial

We designed a metrication infrastructure which consists of two major components (see Figure 2): The supplier's *Collector* collects metrics data through *Data Providers* that, following UNIX philosophy, are implemented as standalone programs/scripts that directly access data sources, or rely on other programs. They generate metric data snippets, and return it in a standardized XML format (see Figure 3). The *Aggregator* component is a database that ingests delivered metric data packages. Data packages can be inserted, analyzed, exported, and, if needed, removed. There is not necessarily a 1:1-relationship between Collector and Aggregator: typically, a customer can collect data from more than one supplier, and can forward the data to higher-tier customers and other stakeholders. Suppliers can run additional own company-internal Aggregators. Data delivery is not designed as automatism but is done manually through encrypted data in

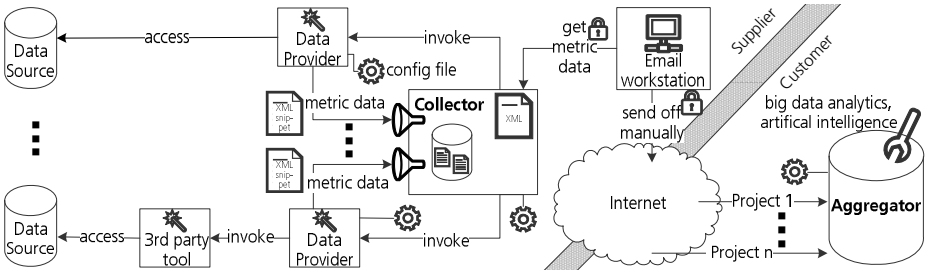


Fig. 2: Software architecture overview of the metrication infrastructure

```
<?xml version="1.0" encoding="utf-8"?>
<xml> <Format version="1.0" />
 <Project name="DemoSat"> <Component name="FlightSW">
  <Delivery milestone="PDR" supplier="OHB-OPF" date="2018-07-16T14:30:36+0200">
   <metric version="1.0" id="A.3.3.01" created="2018-07-16T14:30:36+0200">
    <row> <!--Metric: ECSS-Q-HB-80-04A A.3.3.01 Requirement allocation-->
     <cell description="SSS/TS Requirements traced from SRS">97</cell>
     <cell description="SSS/TS Requirements Total">128</cell>
     <cell description="SRS Requirements traced from SDD">207</cell>
     <cell description="SRS Requirements Total">256</cell>
    </row> ... </metric> ... </Delivery> </Component> ... </Project> </xml>
```

Fig. 3: Data format example of a machine-readable metrics deliverable

(encrypted) email messages so that deliveries can be officially authorized, and senders have control over their sensitive data. Also, using email infrastructure does not require changes to corporate IT-infrastructure/firewalls. AENEAS is currently deployed in one major project, and scheduled to provide data for 18 months, in order to evaluate the concept, the infrastructure and to collect lessons learned. Deployment in a second project is planned.

5 Lessons for Early Metrication Programmes

The following lessons are based on our early experiences with the metrication programme. When introducing metrication, concede anonymity to early projects to mitigate fears of bad consequences. Metrics are hard measurements available to a wider audience that not necessarily understands how to interpret the data, and do not have appropriate background information of the project. Also, initial data is needed to calibrate baselines, and learn how to deal with the data.

ECSS-Q-HB-80-04A metrics definitions appear immature and it is unclear, whether they have been used in practice before. Many descriptions are from our point of view imprecise, ambiguous, or incomplete, although they seem quite detailed. We had to create our own specification to further detail metrics.

Space industry's information system landscape is far from being standardized. So, the main cost driver is adapting Data Providers to data sources for each project. In our experience, effort for implementing data providers can be many

times over what is originally planned. An anecdote may exemplify the problems: Projects are long running (10+ years until launch) and distributed across sites, while parts of the development environment already have heritage of 20 years. For metric *SPR/NCR*[7] *Trend Analysis*, three data providers (instead of one) had to be implemented because of different SPR/NCR information systems.

The price of enabling a metric varies strongly, so efforts are hard to calculate. ECSS-Q-HB-80-04A recommends situation-specific tailoring, taking into account benefits, experience or implementation cost (cf. [4]). Yet, on the other hand, big data analytics works best when generating unexpected hypotheses [21] from collected data, and so speaks against tailoring. This is a strategic decision.

6 Conclusion

Metrication is the foundation of engineering disciplines, and an important part of controlling development. The pressure of modern global economy forces organizations to focus on their performance, be it with respect to quality of products and processes, efficiency, effectiveness, or other attributes. In this world, as Clive Humby noted, "data is the new oil." Although metrics alone are not a universal remedy, they can serve as an early warning system, and, at large, allow the employment of big data analytics and artificial intelligence to improve processes and cooperation for the benefit of involved partners and the industry as a whole.

The AENEAS effort aims to improve mutual understanding in customer-supplier relationships in space projects by increasing the transparency of the software development status through metrication. The 50-year old provocative comparison of "software guys" to the weavers in the tale The Emperor's New Clothes, was found to still have some — polarizing — truth in it. So, practitioners expressed a wish for more transparency, and deemed software metrics/KPIs according to ECSS-Q-HB-80-04A as a viable, partial mitigation.

The presented AENEAS infrastructure consists of two parts that collect metrics data from suppliers and aggregate the delivered data on customer sides. While the AENEAS project has not yet completed its evaluation, we presented some lessons regarding the technical implementation and its use in a real-world, major space project. In the future, with larger amounts of data, we want to be able to answer questions what benefits and costs (individual) metrics have in customer-suppliers relationships. Data may be utilized to enhance metrics, to improve reasoning and planning, to increase cost efficiency, to better tailor metrics to project needs, and to advance process and product quality.

Acknowledgments

The AENEAS project is contracted by the German Aerospace Center on behalf of the German Ministry of Economics and Energy (BMWi) under FKZ 50PS1602. We thank the project teams and participants of our survey.

[7] SPR: Software Problem Report; a mere bug report. NCR: Non-Conformance Report, a severe form with PA/QA and possibly customer involvement.

References

1. Apgar, H.: Space Mission Engineering: The New SMAD (STL vol. 28), chap. Cost Estimating, pp. 289–324. Space Technology Library, Microcosm Press (2011)
2. Baldassari, B.: Squore: A new approach to software project quality measurement. In: Intl. Conf. on Software & Systems Engineering and their Applications (2012)
3. Baldassari, B.: Alambic: An open-source platform for software engineering data management. the case of embedded software development. In: ICSSEA (2016)
4. Basili, V.R., Lindvall, M., Regardie, M., et al.: Linking software development and business strategy through measurement. Computer **43**(4), 57–65 (2010)
5. Basili, V.R., McGarry, F.E., Pajerski, R., Zelkowitz, M.V.: Lessons learned from 25 years of process improvement: The rise and fall of the nasa software engineering laboratory. In: 24th Intl. Conf. on Software Engineering. pp. 69–79. ACM (2002)
6. Boehm, B.: Software and its impact: A quantitative assessment. Tech. rep. (1972)
7. Boehm, B.: Get ready for agile methods, with care. Computer **35**, 64–69 (2002)
8. Donaldson, S.E., Siegel, S.G.: Successful Software Development. P.-Hall (2001)
9. Ebert, C., Dumke, R.: Software Measurement: Establish-Extract-Evaluate-Execute. Springer Science & Business Media (2007)
10. ECSS-E-HB-40-01A: Space engineering – agile software development handbook. Standard, ECSS Secretariat, ESA ESTEC (2018 (to appear))
11. ECSS-E-ST-40C: Space engineering – software. Standard, ECSS Secretariat (2009)
12. ECSS-Q-HB-80-04A: Space product assurance – software metrication programme definition and implementation. Standard, ECSS Secretariat, ESA ESTEC (2011)
13. ECSS-Q-ST-80C: Space product assurance – software product assurance. Standard, ECSS Secretariat, ESA ESTEC (2009)
14. Gennen, K.: Auswirkungen hybrider projektvorgehensmethoden auf den softwareerstellungsvertrag. In: Engstler, M., et al. (eds.) PVM. GI, Bonn (2016)
15. Greves, D., Schreiber, B., Maxwell, K., et al.: The esa initiative for software productivity benchmarking and effort estimation. esa bulletin **87**, 84–88 (1996)
16. IEEE standard glossary of software engineering terminology (1990)
17. ISO/IEC 15939: Software engineering – software measurement process. Standard, International Organization for Standardization (2007)
18. Klünder, J., et al.: Towards understanding the motivation of german organizations to apply certain software development methods. In: 3rd HELENA. Springer (2018)
19. Kuhrmann, M., et al.: Hybrid software and system development in practice: Waterfall, scrum, and beyond. In: Int. Con. Software and System Process. IEEE (2017)
20. Lincke, R., Lundberg, J., Löwe, W.: Comparing software metrics tools. In: International Symposium on Software Testing and Analysis. pp. 131–142. ACM (2008)
21. Müller, I.: Big data analytics. Lecture, OpenHPI (Nov 2017)
22. Prause, C.R., Bibus, M., Dietrich, C., Jobi, W.: Managing Software Process Evolution for Spacecraft from a Customer's Perspective, pp. 137–163. Springer (2016)
23. Prause, C.R., Reiners, R., Dencheva, S.: Empirical study of tool support in highly distributed research projects. In: Intl. C. on Glob. Soft. Eng. ICGSE, IEEE (2010)
24. Prause, C.R., Werner, J., Hornig, K., Bosecker, S., Kuhrmann, M.: Is 100% test coverage a reasonable requirement? lessons learned from a space software project. In: Intl. Conf. on Product-Focused Software Process Improvement. PROFES (2017)
25. Rechtin, E.: Reducing the costs of space science research missions. In: Proceedings of a Workshop. pp. 23–29. National Academy Press (1997)
26. Reddy, V.R.: Software Metrics Tool. Master's thesis (2016)
27. Scheibel, W., Weyand, C., Döllner, J.: Evocells – a treemap layout algorithm for evolving tree data. In: Int. Conf. on Inform. Vis. Theory and Applications (2018)