Good RE Artifacts? I Know It When I Use It!

Henning Femmer
Technische Universität München
femmer@in.tum.de

Andreas Vogelsang
Technische Universität Berlin
andreas.vogelsang@tu-berlin.de

I. Motivation and Context

The definition of high-quality or good RE artifacts is often provided through normative references, such as quality standards or text books (e.g., ISO/IEEE/IEC-29148 [1]). We see various problems of such normative references.

Quality standards are incomplete. Several quality standards describe quality through a set of abstract criteria. When analyzing the characteristics in detail, we see that there are two different types of criteria: Some criteria, such as ambiguity, consistency, completeness, and singularity are factors that describe properties of the RE artifact itself. In contrast, feasibility, traceability and verifiability state that activities can be performed with the artifact. This is a small, yet important difference: While the former can be assessed by analyzing just the artifact by itself, the latter describe a relationship of the artifact in the context of its usage. Yet this usage context is incompletely represented in the quality standards: For example, why is it important that requirements can be implemented (feasible in the terminology of ISO-29148) and verified, but other activities, such as maintenance, are not part of the quality model? Therefore, we argue that normative standards do not take all activities into account systematically, and thus, are missing relevant quality factors.

Quality standards are only implicitly context-dependent. One could go even further and ask about the value of some artifact-based properties such as singularity. A normative approach does not provide such rationales. This is different for activity-based properties, such as verifiability, since these properties are defined through their usage: If we need to verify the requirements, properties of the artifact that increase verifiability are important. If we do not need to verify this requirement, e.g., because we use the artifacts only for task management in an agile process, these properties might not necessarily be relevant. This example shows that, in contrast to the normative definition of quality in RE standards, RE quality usually depends on the context.

Quality standards lack precise reasoning. For defining most of the aforementioned criteria, the standards remain abstract and vague. For some criteria, such as ambiguity, the standards provide a detailed lists of factors to avoid. However, these criteria have an imprecise relation to the abstract criteria mentioned above, and, consequently, the harm that they might potentially cause remains unclear.

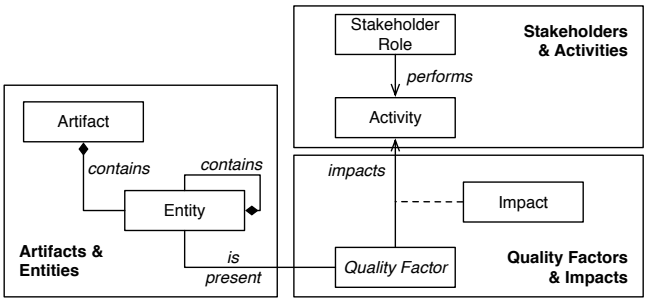

Fig. 1. The ABRE-QM meta model, based on [2]

II. Proposed Solution

We postulate that creating an RE artifact is rarely an end in itself, but just a means to understand and reach the project's goals. Following this line of thought, the purpose of the requirements artifact is to support the stakeholders in whatever activities they are performing in the project. This change of view means that it is unreasonable to talk about good or bad RE artifacts in general. What is good and what is bad must always be assessed with respect to the given context and, more specifically, with respect to the activities that are conducted with the RE artifacts. In fact, we argue that common quality criteria, even completeness and correctness, have to be rethought from a quality in use perspective. This contributes a novel view on requirements engineering artifact quality, which discusses RE artifact quality from a quality-in-use viewpoint. To express this view, we contribute an activity-based RE quality meta model that contains the following elements (see Fig. 1):

- An **artifact** is a file or file-like representation of a product of an RE endeavour, such as a use case document.
- An **entity** is a logically coherent part of an artifact or another entity, such as a use case or a step in the flow of a use case.
- A **stakeholder role** is a role with an interest in the project [3], such as the tester.
- An **activity** is an invested effort, which involves one or more of the aforementioned artifacts and stakeholders, such as creating test cases.
- A **quality factor** is a property that is or is not present in an entity. This property must be objectively assessable to be used during QA. Furthermore, each quality factor consists of a set of impacts.
- An **impact** is an explicit relation between a quality factor and an activity. The impact influences either *effectiveness* or *efficiency* of that activity.

This model enables researchers to provide practitioners with a precise definition of what they consider to be bad quality, why (i.e., because of which consequences) and in which context (i.e., under the assumption that which activities are performed). Practitioners can then use such a precise quality model and, based on artifacts, activities and impacts, decide which quality characteristics are relevant for their context.

III. So What?

We have applied the proposed meta model for different purposes. The meta model proved beneficial in several contexts that are discussed in the following.

Activity-based RE guidelines. Many companies nowadays have guidelines to help employees improve their requirements and to create a baseline for quality. However, as stated before, these guidelines are often incomplete and imprecise. We argue that guidelines that are defined in an activity-based manner could help in this direction. In a first study [4], practitioners reported that a translated guideline helps to both discuss validity of the existing rules and to create more complete guidelines.

Activity-based cost estimation. We have used the proposed meta model to develop cost models to enable an informed decision making process. In a recent study [5], we used an instance of the meta model to characterize the cost and benefits of refactoring functional parts that reoccur in several functions of a system specification. The decision whether a refactoring pays off heavily depends on the context in which the respective system specification is used. To take this context into consideration, we identified activities that are performed with the system specification, and we identified cost factors that affect these activities in the original and the refactored version. Cost factors are a specific form of quality factors as apparent in the meta model. As a result, the decision whether to refactor a specification or leave it as it is can be assessed with respect to the usage context. For one context, the refactoring pays off because functions must be tested frequently and recurring parts in the functions makes testing more expensive. For another context, the refactoring does not pay off because the responsibility for implementing the refactored parts is in a different department and, thus, we need to consider costs for knowledge transfer. A similar approach is taken by Hauptmann et al. [6] to decide when and if test step automation pays off.

Activity-based tailoring of Requirements Templates. Requirements templates are blueprints that determine the syntactic structure of a single requirement. One reported advantage of requirements templates is that they facilitate *complete* specifications of requirements. However, what complete actually means depends on how the requirement is used. The information that needs to be provided by a requirement is determined by the activities that are performed based on the requirement. For example, when performance tests will be conducted for the system, the requirement templates must enforce to augment requirements with detailed information about the desired reaction times and assumed conditions [7]. We have shown how activity-based models can be used to tailor requirements templates in a way that the information they demand for a requirement fit the actual usage in a specific development context [8]. The result is a set of requirements templates that are more specific and expressive than the general templates that are proposed to fit every situation.

Activity-based quality assurance. The presented paradigm also has strong implications on quality assurance. This is both for constructive aspects, such as tailoring guidelines to requirements use, but also analytical approaches, such as requirements smells [4].

Activity-based impact of RE quality and a generic ABRE-QM. Lastly, the paradigm helps to steer and unite research by providing it with a common theory: Research can be structured along quality factors and focus on which activities are impacted by the quality factor. That way, both defining the quality factor and understanding its impact is precise and structured, as we did, for example, in our experiment on the impact of passive voice on understanding [9].

If the research continues along this theory, this will inevitable lead to a generic ABRE-QM, which resembles existing knowledge on RE artifact quality and its impact.

IV. Conclusions

The strengths of our approach, as reported by practitioners [2], lies in the clarity of the reasoning. Taking activities as the basis provides with a simple rule whether or not something is of better or worse quality: *If it hinders someone, it is bad quality*. This comes with the same strength, that this rule at the same time generates hypothesis for each postulated rule for good or bad quality. We argue that this quality model enables research to both argue more clearly about their results, but also conduct better studies, with a clearer research focus.

References

[1] ISO/IEC/IEEE, "Systems and software engineering – Life cycle processes – Requirements engineering," International Organization for Standardization, Geneva, Switzerland, ISO/IEC/IEEE 29148:2011(E), 2011.

[2] H. Femmer, J. Mund, and D. Méndez Fernández, "It's the activities, stupid!: A new perspective on RE quality," in *International Workshop on Requirements Engineering and Testing (RET)*, 2015.

[3] K. Pohl, *Requirements engineering: fundamentals, principles, and techniques*. Springer Publishing Company, Incorporated, 2010.

[4] H. Femmer, D. Méndez Fernández, S. Wagner, and S. Eder, "Rapid quality assurance with requirements smells," *Journal of Systems and Software*, 2016.

[5] A. Vogelsang, H. Femmer, and M. Junker, "Characterizing implicit communal components as technical debt in automotive software systems," in *Working Conference on Software Architecture (WICSA)*, 2016.

[6] B. Hauptmann, M. Junker, S. Eder, C. Amann, and R. Vaas, "An expert-based cost estimation model for system test execution," in *International Conference on Software and System Process (ICSSP)*, 2014.

[7] J. Eckhardt, A. Vogelsang, H. Femmer, and P. Mager, "Challenging incompleteness of performance requirements by sentence patterns," in *International Requirements Engineering Conference (RE)*, 2016.

[8] J. Eckhardt, A. Vogelsang, and H. Femmer, "An approach for creating sentence patterns for quality requirements," in *International Workshop on Requirements Patterns (RePa)*, 2016.

[9] H. Femmer, J. Kucera, and A. Vetrò, "On The Impact of Passive Voice Requirements on Domain Modelling," in *International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2014.