Henning Femmer, Andreas Vogelsang

Requirements Quality Is Quality in Use

Femmer, H.; Vogelsang, A. (2019). Requirements Quality Is Quality in Use. IEEE Software, 36(3), 83–91. https://doi.org/10.1109/ms.2018.110161823

Requirements Quality is Quality in Use

Henning Femmer and Andreas Vogelsang

***Abstract*—The quality of requirements engineering artifacts is widely considered a success factor for software projects. Currently, the definition of high-quality or good RE artifacts is often provided through normative references, such as quality standards, textbooks, or generic guidelines. We see various problems of such normative references: (1) It is hard to ensure that the contained rules are complete, (2) the contained rules are not context-dependent, and (3) the standards lack precise reasoning why certain criteria are considered bad quality. To change this understanding, 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 an RE artifact is to support the stakeholders in whatever activities they are performing in the project. This purpose must define high-quality RE artifacts. To express this view, we contribute an activity-based RE quality meta model and show applications of this paradigm. Lastly, we describe the impacts of this view onto research and practice.**

***Index Terms*—Quality, quality standards, requirements, documentation, roadmap**

Three Actionable Insights:

- Always remember: Requirements engineering artifacts are *a means, not an end.*
- Therefore, before writing your requirements, think about *the readers* and *how they use the artifacts* first.
- Use this simple model to define who is using RE artifacts, what RE artifacts are used for, and how RE artifacts should therefore look like.

I. Current Standards are Incomplete, Inadequate and Imprecise on Requirements Quality.

Requirements Engineering (RE) artifacts are central entities in the software engineering process. Based on these artifacts, project managers estimate effort, designers create architectures, developers build the system, and test managers set up a test-strategy. Consequently, quality defects in RE artifacts can cause expensive consequences in subsequent software development activities. Therefore, quality control of RE artifacts is key for successful software development projects.

The definition of high-quality or good RE artifacts is often provided through normative references, such as quality standards or textbooks (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 an 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 consider all activities systematically, and thus, are missing relevant quality factors.

Quality standards are not context-dependent. One could go even further and ask about the value of some artifact-based properties such as singularity or formality. Still widely cited quality models of the past [2] proclaimed that (all) projects should strive towards formalized requirements. What is the purpose and reason behind such a property? A normative approach does not provide rationales. This is different for activity-based properties, such as verifiability, since these properties are defined by their usage: If we need to verify the requirements, properties of the artifact that increase verifiability are important. In particular, we need to understand up-front how we want to verify the requirements. For a formal verification, formalized requirements are a reasonable approach. For manual testing, however, formalized requirements might actually make them harder to understand and, therefore, harder to test. This example shows that, in contrast to the normative definition of quality in RE standards, RE quality usually depends on the usage context.

Quality standards lack precise reasoning. The standards remain abstract and vague when defining the criteria. Only for some criteria, such as ambiguity, the standards provide a detailed list of factors to avoid (e.g., comparatives). However, even then, the concrete impact of these factors (such as comparatives) onto the abstract criteria (such as ambiguity) remains unclear.

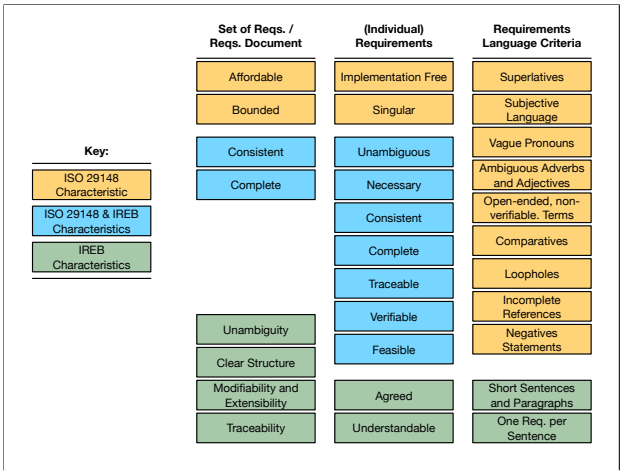

Fig. 1: This figure compares the quality characteristics of ISO 29148 and the IREB syllabus. Blue characteristics are shared characteristics, orange and green characteristics appear only in one of the standards.

II. Sidebar: Comparison of RE quality standards

To get a taste of current RE quality standards, we compare the ISO/IEC/IEEE-29148 [1] quality standard with the definition of quality attributes from the curriculum of the International Requirements Engineering Board (IREB), a certification also widely used in industry.

Both standards define a quality model through a simple list of characteristics. According to the standards, good requirements documents are those in which these characteristics are present. The standards share nine of the characteristics (see Fig. 1), mostly those characteristics defined in earlier literature and standards, such as the IEEE 830. However, the standards disagree on more characteristics than they agree on. In particular, the standards disagree when it comes to concrete language criteria. Unfortunately, even when the standards agree on the characteristics, the definitions of these characteristics vary in specific details. Take, for example, consistency. While the IREB definition only considers disagreeing requirements as inconsistent, the ISO 29148 definition of inconsistency also includes duplication issues and terminological deficiency. In addition, the IREB assesses quality characteristics on a continuous scale, whereas the definitions of the ISO standard suggest a Boolean interpretation.

At a glance, both standards share the same approach towards quality, but their details differ tremendously. This is especially true for the concrete, assessable language criteria. We argue that these differences indicate two problems. First, the missing agreement on the level of concrete language criteria indicates that we do not yet know what is good or bad quality, and that we have little to no established understanding of the impacts of concrete language criteria. Second and even more problematic, the missing agreement at the level of abstract quality characteristics indicates there is neither an established understanding about nor an established approach towards quality for RE artifacts as a whole.

Other authors have also noticed these deficiencies and came up with quality models that consider the viewpoint of the user. For example, Lindland et al. [3] define different facets of quality (syntactic, semantic, pragmatic), where pragmatic quality emphasizes the extent to which the requirements model is understood by the user. Deissenboeck et al. define an activity-based quality model [4] for quality-in-use characteristics of source code, such as maintainability. Our work details the pragmatic quality facet through the use of activity-based quality models.

III. Goals of Requirements Engineering

Let us take a step back. If we want to get to the bottom of RE artifact quality, we need to reconsider the goals of requirements engineering itself since RE artifacts should eventually support the goals of RE. Following the definitions of the goals of RE as understood by Glinz [5, p.18], we understand quality in RE as the degree to which the following goals are sufficiently fulfilled for system stakeholders and the project team:

① **Understand stakeholders' needs**: High quality in RE is the degree of correct and complete understanding of the goals, expectations, and constraints of the system stakeholders.

② **Achieve agreement**: High quality in RE is the degree of agreement on a system that manifests the consensus of all system stakeholders. To this end, high quality in RE correctly prioritizes requirements, and ensures that a best-possible solution is derived for the system stakeholders' needs (iteration between problem and solution space, see twin-peaks model [6]).

③ **Create the same mental model between all system stakeholders**: High quality in RE is the degree to which these system stakeholders' needs and the derived consensus are communicated correctly and completely between all involved system stakeholders in the project.

④ **Structure & manage requirements-based activities**: Many project activities are structured along the system stakeholders' needs, e.g., in the form of requirements. Some exemplary activities are estimating the costs, planning the implementation of the system, developing the system, or testing the system. Consequently, high quality in RE is the degree to which engineers working with the requirements (i.e., the information) can efficiently and effectively use the requirements to execute their requirements-based activities [7].

RE artifact quality should be defined and assessed with respect to the achievement of these goals.

IV. A Different View on Requirements Quality

Gross and Doerr [7] argue that in order to create high-quality requirements documentations that fit the specific demands of successive document stakeholders, the research community needs to better understand the particular information needs of downstream development roles. We follow this idea but relate artifact quality to development activities and not to document stakeholders because information needs of stakeholders may change depending on the activities they have to perform in a specific context.

In the following, we describe our approach towards requirements artifact quality. We first describe the basic concepts, then the detailed model, and finally an exemplary quality factor.

A. The Idea

We postulate that creating an RE artifact is rarely an end in itself, but just a means to reach the project's goals. In particular, they are a tool to reach the goals of RE, as described in the previous section. Following this line of thought, the purpose of a 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. More specifically, good quality depends on the RE artifact stakeholders and the activities that they conduct 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.

B. The Model

To define RE artifact quality, we designed Activity-Based RE artifact Quality Models (ABRE-QMs). To describe the structure of ABRE-QMs, we provide an ABRE-QM meta model that introduces the concepts needed to describe an ABRE-QM (see Fig. 2 left):

An artifact is a documented collection of requirements entities, which is produced during an RE process. An example for an artifact is a *use case document*.

An entity is a coherent documented information. An entity can be an information content item, but can also be further decomposed, for example into the linguistic components of such a content item. Examples for entities are a *use case*, an *alternative flow*, or a *step* within the flow.

A stakeholder role is the role of someone with an interest in the RE artifact [8], such as a test engineer. Each role can include other roles that are more generic. For example, both *test engineer* and *developers* are also *readers* of the requirements artifact. Therefore, quality factors that affect the activity *read*, affect all *readers* of the artifact, including *test engineers* and *developers* through their included generic role *reader*. This allows combining shared activities that multiple stakeholders must execute.

An activity is an invested effort, which involves one or more of the aforementioned artifacts, such as creating test cases, and one or more of the aforementioned stakeholder roles, such as the *test engineer*. An activity can be broken down into subactivities. For example, the testing activity is decomposed into *creating*, *running*, and *maintaining* test cases.

A quality factor is a property that is or is not present in an entity. This property must be objectively assessable through a measure to be used for quality control.

An impact is an explicit relation between a quality factor and an activity. The impact influences either *effectiveness* or *efficiency* of that activity. This impact is explicitly discussed through: First, a reason, i.e., an argumentation why the presence of a specified characteristic (the quality factor) of an artifact impacts the associated activity; second, consequences on costs, schedule or quality of the developed system; and third, a source from which this impact was derived and which can provide further information, i.e., a requirements quality standard or corporate guidelines.

A context factor influences the impact of a quality factor. For example, the problematic impact of a *passive voice* requirement varies, depending on the background of the reader. If the reader has no or few domain knowledge, the passive voice has a stronger impact. In contrast, in cases where the reader is well aware of the domain and the ideas of the system, the impact can be less problematic. Context factors can be human, process or tool factors.

An assessment is a description for evaluating an entity against a quality factor. The application of an assessment against an entity results in a (potentially empty) set of quality defects (cf. [4]). We see three potential categories of assessments: manual, automatic, and semi-automatic assessments.

C. An Exemplary Quality Factor

To foster understanding, this section provides an exemplary excerpt of an ABRE-QM (see Fig. 2 right). The example shows the definition of one quality factor, namely the presence of explicit steps in a use case flow.

1. Artifacts and entities: A *use case document* is a common artifact for specifying functional requirements to software systems. A use case usually contains a *basic flow*, which is a sequence of steps that describes how the user interacts with the system.

2. Stakeholder roles: For the sake of simplicity, in this example, we consider only *test engineers*.

3. Activities: When we analyze how a test engineer processes the use case document in a specific project, we discover that, among other activities, the test engineers goes through the use case steps and *creates test step(s)* for the use case's basic flow.

4. Quality factors: It is considered good practice in use cases to *explicitly separate* each step instead of describing the whole basic flow in one text block. With the aforementioned context and activity in mind, we understand why a use case with this quality factor is considered higher quality: The test engineer can directly translate the use case steps to test steps. Therefore, the test engineer's task of creating a test sequence can be executed more effectively (and maybe also more efficiently) when the factor is present in the use case. Fig. 2 explicates this reasoning through a positive ('+') impact in an ABRE-QM. Please note, that for simplicity, we only discuss one of the impacts of this quality factor here.

5. Context factors: One could consider the applied tool environment to be a context factors. Depending on the concrete tools in use, the translation is more or less efficient.

6. Assessment: One could discuss various types of assessments, depending on the tools used. An easy-to-apply assessment is a manual review, which can spot this quality defect. In addition, for various requirements management tools, one could discuss automatic (or at least semi-automatic) methods through automatic analysis of the use case's structure.

This example shows the definition of one quality factor. An ABRE-QM is a composition of a set of such quality factors

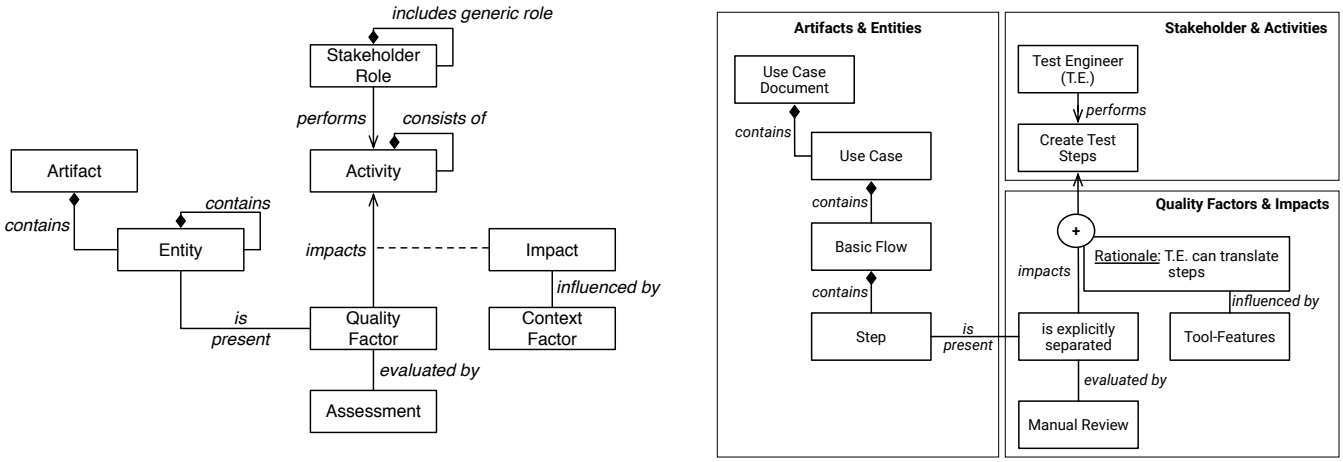

Fig. 2: This figure shows the ABRE-QM meta model (left) and a simple example of a quality factor in an ABRE-QM (right). The meta model consists of artifacts and their decomposition into entities, quality factors and their impact on activities, which are performed by certain stakeholder roles. Impacts are influenced by context factors. Lastly, quality factors are evaluated by assessments. The example discusses why explicitly separated steps in basic flows of use cases are considered good quality. In this example, we discuss the impact on creating test steps, i.e., explicitly separated steps in basic flows allow more efficient and effective creation of test steps through reuse.

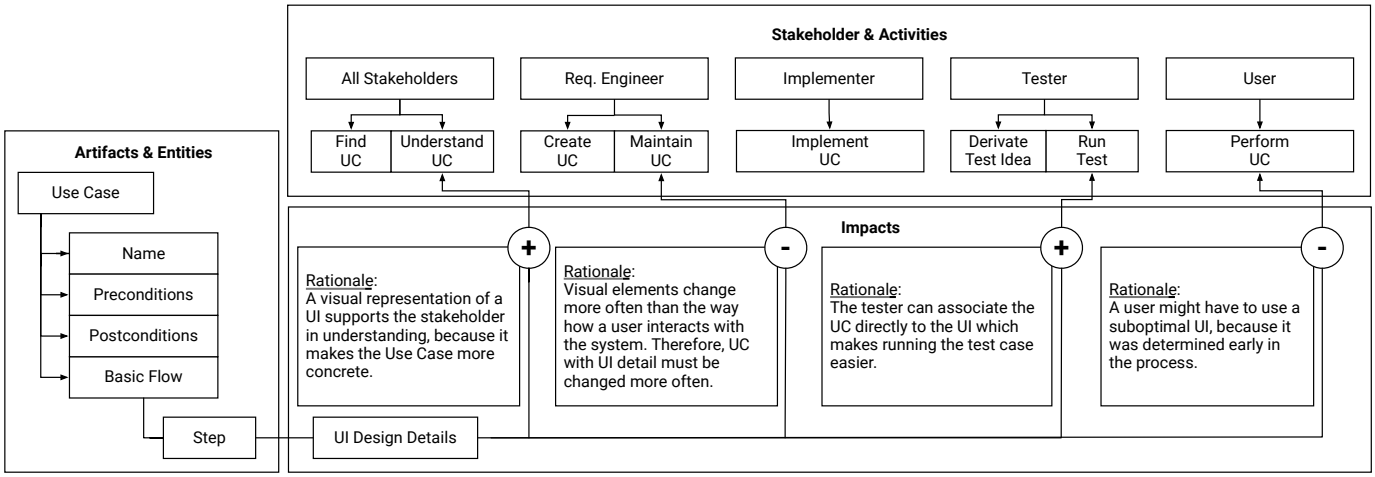

Fig. 3: This figure shows a more complex ABRE-QM. It discusses positive as well as negative impacts of UI design details in use cases: On the one hand, UI details in use case descriptions eases understanding the use case and running tests. On the other hand, it makes maintaining the use cases less efficient and might lead to suboptimal design, which makes it harder for the user to perform the use case.

with their respective relations. Fig 3 shows a more extensive example, where a quality factor has positive and negative impacts onto activities and needs to be evaluated in context.

This model enables researchers to provide practitioners with a precise definition of what they consider to be good or bad quality, why (i.e., due to which consequences) and in which context (i.e., based on which activities). 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.

V. Applications in Research and Practice.

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. Nowadays, many companies have generic guidelines to help employees improve their requirements and to create a baseline for quality. We argue that guidelines that are defined in an activity-based manner could help to make these guidelines more complete, precise, and specific for their context. In a first study [9], practitioners reported that a translated guideline helps to both discuss validity of the existing rules and to create guidelines that are more complete.

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 requirements

specifications that are more *complete*. However, what complete actually means depends on how requirements are used: The information that needs to be provided in a requirement is determined by the activities that are performed based on the requirement. In a recent paper [10], we used activity-based models to tailor requirements templates in a way that the information they demand for a requirement fit the actual usage in a specific development context. The result is a set of requirements templates that are more specific and expressive than general templates that are proposed to fit every situation.

Activity-based Cost Estimation. We used the proposed meta model to develop cost models to enable an informed decision making process: In a recent study [11], 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. Therefore, 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 present 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.

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 [9], where ABRE-QMs enabled us to decide which quality characteristics should hold in a certain context.

Activity-based Impact of RE Quality in a Common Theory. Lastly, the paradigm helps steer and unite research by providing it with a common theory: Research can be structured along quality factors and thus focus on activities that are impacted by a certain quality factor. That way, both defining the quality factor and understanding its impact follows a precise structure. We followed this example in our experiment on the impact of passive voice on understanding requirements [12].

VI. So What?

Based on our activity-based model, we can categorize where the research in the field should be heading.

A. What Should Practitioners Do?

If you are a practitioner trying to improve the RE process in your company by increasing the artifact quality, we argue that these steps help to create more efficient and effective requirements artifacts:

1) Always remember: Requirements engineering artifacts are *a means, not an end.*
2) Therefore, before writing requirements, we suggest to think about *the readers* and *how they use the artifacts*: Which information do they need? Afterwards, we advise to create a model of the RE artifacts produced in the company, the stakeholders that use these artifacts, and the activities that the stakeholders perform. The meta model presented in this article can help to structure this model.
3) What is most important in our opinion is to talk to the stakeholders who use an artifact to assess what helps or hinders them in performing their tasks. Their experience can be included as quality factors to the quality model.
4) After the basic model of artifacts, activities, and quality factors is established, one can start introducing company-wide guidelines or quality assurance measures based on this model. We argue that these guidelines and measures are more complete, specific, and justified than normative references because they have a direct relation to activities that profit from them. Requirements patterns may help to remind the engineers of the information they need to document (see [10]).
5) Lastly, it is important to evolve and maintain this model over time. The stakeholders, activities, and therefore also the notion of artifact quality may change over time or there might be new quality factors that should be added. Additionally, a change in some context factors (e.g., tooling improvements) may mitigate negative impacts. This could mean that maintaining some quality factors may no longer be relevant.

Implementing these steps in a company takes some effort. However, we see this as the most promising approach to effective quality definitions that are accepted by employees because they see the benefit of following the guidelines. If companies already maintain a reference model of artifacts and stakeholders, the initial effort may be less compared with companies that first have to create such a model. To reduce initial costs, a company may focus on the most crucial (or cost-intensive) activity and start building an ABRE-QM with guidelines for this activity. As a result, the activity will directly benefit from the new quality factors. Additional activities and artifacts may successively be added to the model. We suggest defining the role of a quality manager who is responsible for creating and maintaining the ABRE-QM.

B. What Is Left For Research?

The activity-based approach for quality definitions strongly benefits from a unified and well-tested body of quality factors and related impacts. If research continues along this theory, the community can create a generic ABRE-QM, which will resemble the existing knowledge on RE artifact quality. The precision of such a theory would allow researchers to discuss results in the field systematically and its focus on activities would enable practitioners to understand and weigh consequences of bad quality in a structured manner. In the long run, this paradigm could even be extended beyond artifacts to create a general RE quality theory.

To accomplish this vision, researchers should work on the following topics:

1) Create a **reference artifact and usage model** that serves as a list of typical stakeholders, their most important activities, and typical artifacts that are used in the activities. Practitioners may use this reference model as a starting point for a company-specific model.

2) Create a **taxonomy of quality factors** that serves as a body of knowledge of quality factors. To observe these, one can look through produced artifacts, for example:
   - Review protocols that indicate the effort that was invested during QA of the RE artifacts (as we, for example, did in an earlier paper [13])
   - Incorrect test cases or incorrect test results that show that the test case engineer misunderstood the RE artifacts
   - Requirements change requests, or defects in bug tracking systems that can be traced back to RE artifacts, to understand defects in the RE artifact that are discovered during development or further activities
   - Concrete changes that have been performed on the RE artifacts to understand maintenance efforts (as for example performed by Basirati et al. [14])
3) Create a **taxonomy of impacts** that provides a list of well-examined effects of quality factors on activities. To evaluate impacts, one may use interviews [15], case studies [14], or experiments [12].

Practitioners will benefit from this community effort only if the necessary effort for tailoring the reference model and taxonomies is reasonable. This should be investigated in future work. Another issue that researchers should examine is how to handle conflicts in the quality model: A quality factor that has a positive impact onto one activity may have a negative impact onto another activity (see Fig. 3).

VII. Conclusions

The strength of our approach, as first interviews with practitioners indicate [15], lies in its clarity of reasoning. Taking activities as the basis provides a simple rule whether or not something is of better or worse quality: *If it hinders someone, it is bad quality*. This rule, at the same time, generates falsifiable hypotheses 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. We furthermore argue that this model provides practitioners with a precise and valid approach to understand: What are good requirements artifacts in my case?

Acknowledgements

This work was performed within the project Q-Effekt; it was funded by the German Federal Ministry of Education and Research (BMBF) under grant no. 01IS15003 A-B. The authors assume responsibility for the content.

PLACE PHOTO HERE

Henning Femmer Henning Femmer holds a Ph.D. from Technical University Munich (TUM) and is a co-founder of the requirements engineering consulting company Qualicen. His research focuses on improving the efficiency and effectiveness of requirements quality control, with a particular focus on automatic methods. He publishes at academic venues, such as ICSE, RE, ESEM, but also speaks at industry-focused events, such as REConf or Embedded World. In both his research and practical work he aims to combine scientific rigor with industrial applicability in order to efficiently deliver high quality.

PLACE PHOTO HERE

Andreas Vogelsang Andreas Vogelsang is a professor for systems engineering at the Berlin Institute of Technology (TU Berlin). He closely cooperates with the Daimler Center for Automotive IT Innovations (DCAITI) by leading the software engineering research group. His research interests comprise model-based requirements engineering, requirements specification quality, and software architectures for software-intensive systems. He has published over 20 publications in international journals, conferences, and workshops including ICSE, RE, and REFSQ. Additionally, he participated in several research collaborations with industrial partners especially from the automotive domain. Most of the research collaborations focused on how to create high-quality requirements artifacts and how to improve the RE process.

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] K. Pohl, "The three dimensions of requirements engineering," in *International Conference on Advanced Information Systems Engineering (CAiSE)*, 1993.

[3] O. I. Lindland, G. Sindre, and A. Solvberg, "Understanding quality in conceptual modeling," *IEEE Software*, vol. 11, 1994.

[4] F. Deissenboeck, S. Wagner, M. Pizka, S. Teuchert, and J.-F. Girard, "An activity-based quality model for maintainability," in *International Conference on Software Maintenance and Evolution (ICSM)*, 2007.

[5] M. Glinz, "A glossary of requirements engineering terminology," International Requirements Engineering Board and University of Zurich, Tech. Rep., 2014.

[6] B. Nuseibeh, "Weaving together requirements and architectures," *IEEE Computer*, vol. 34, no. 3, 2001.

[7] A. Gross and J. Doerr, "What you need is what you get!: The vision of view-based requirements specifications," in *Requirements Engineering Conference (RE), 2012 20th IEEE International*, 2012.

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

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

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

[11] 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.

[12] 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.

[13] A. Vogelsang, H. Femmer, and C. Winkler, "Take care of your modes! An investigation of defects in automotive requirements," in *International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ)*, 2016.

[14] M. R. Basirati, H. Femmer, S. Eder, M. Fritzsche, and A. Widera, "Understanding changes in use cases: A case study," in *International Requirements Engineering Conference (RE)*, 2015.

[15] 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.