Requirements Engineering (2023) 28:507–520
https://doi.org/10.1007/s00766-023-00405-y

VIEW POINT

Requirements quality research: a harmonized theory, evaluation, and roadmap

Julian Frattini[1] · **Lloyd Montgomery**[2] · **Jannik Fischbach**[3,4] · **Daniel Mendez**[1,4] · **Davide Fucci**[1] · **Michael Unterkalmsteiner**[1]

Received: 29 September 2022 / Accepted: 10 July 2023 / Published online: 12 August 2023

Abstract
High-quality requirements minimize the risk of propagating defects to later stages of the software development life cycle. Achieving a sufficient level of quality is a major goal of requirements engineering. This requires a clear definition and understanding of requirements quality. Though recent publications make an effort at disentangling the complex concept of quality, the requirements quality research community lacks identity and clear structure which guides advances and puts new findings into an holistic perspective. In this research commentary, we contribute (1) a harmonized requirements quality theory organizing its core concepts, (2) an evaluation of the current state of requirements quality research, and (3) a research roadmap to guide advancements in the field. We show that requirements quality research focuses on normative rules and mostly fails to connect requirements quality to its impact on subsequent software development activities, impeding the relevance of the research. Adherence to the proposed requirements quality theory and following the outlined roadmap will be a step toward amending this gap.

Keywords Requirements quality · Theory · Survey

1 Introduction

The empirical evidence of the impact of requirements engineering (RE) on the software development life cycle has shown that the quality of requirements artifacts and processes influences project success and budget adherence [1–3]. Moreover, the cost of defects introduced during the RE phase of a project is reported to scale exponentially the longer they remain undetected [4]. This necessitates quality assurance techniques capable of detecting RE defects as soon and as reliably as possible.

Requirements quality research is dedicated to supporting the software engineering process with the means to evaluate and improve the quality of requirements, mainly focusing on requirements artifacts [5]. However, recent systematic investigations of requirements quality literature revealed a lack of rigor and relevance of these contributions [6, 7]. Moreover, the impact of the quality factors proposed in the literature (i.e., requirements writing rules) remains largely unexplored in practice [7], hindering its adoption in industry [8–11].

Existing quality theories and frameworks are too abstract to guide requirements quality research at an operational level [12, 13]. These theories often only divide quality into

✉ Julian Frattini
Julian.Frattini@bth.se

Lloyd Montgomery
lloyd.montgomery@uni-hamburg.de

Jannik Fischbach
jannik.fischbach@netlight.com

Daniel Mendez
Daniel.Mendez@bth.se

Davide Fucci
Davide.Fucci@bth.se

Michael Unterkalmsteiner
Michael.Unterkalmsteiner@bth.se

[1] Blekinge Institute of Technology Valhallavägen 1, 37140 Karlskrona, Sweden

[2] University of Hamburg Mittelweg 177, 20148 Hamburg, Germany

[3] Netlight Consulting GmbH Sternstraße 5, 80538 Munich, Germany

[4] Fortiss GmbH Guerickestrasse 25, 80805 Munich, Germany

sub-categories without any means of applicability. In this paper, we argue for the need for a theoretical and operationalizable foundation of requirements quality research. We review the closely related software quality research and draw parallels to requirements quality research to consolidate a harmonized requirements quality theory. Additionally, we survey requirements quality literature with respect to the theory to reveal current shortcomings. Accordingly, we make the following contributions:

1. A harmonized requirements quality theory serving as a theoretical foundation for requirements quality research.
2. A survey of requirements quality research revealing if and how concepts of the theory are reported in the state of the art, but also emphasizing shortcomings.
3. A consequent research roadmap aimed at mitigating these shortcomings.

The rest of this manuscript is organized as follows: Sect. 2 illustrates the evolution of software quality research and draws the parallel to requirements quality research. In Sect. 3, we derive a harmonized requirements quality theory from this comparison. This theory is used to evaluate the state of requirements quality research in Sect. 4 and reveal current shortcomings. The consequent research roadmap to mitigate these shortcomings is presented in Sect. 5 before concluding in Sect. 6.

2 Software quality research

Software quality research follows a similar premise as requirements quality research. It is necessary to control the quality of software artifacts (e.g., source code) as it impacts the overall quality of the development life cycle and the final product. This premise aligns with the aim of requirements quality research. To show commonalities and differences between these two research fields, we review the evolution of software quality research in Sect. 2.1 and draw a parallel to requirements quality research in Sect. 2.2. We reach conclusions about the necessary direction the latter needs to take.

2.1 Evolution of software quality research

Software quality research revolves around assessing the quality of software artifacts [14]. In the following, we describe the evolution of the field according to Broy et al. [14] and Deissenboeck et al. [15].

Guidelines and Metrics-based approaches Guidelines are the simplest approach for controlling the quality of software artifacts. For example, the Java coding conventions [16] prescribe—among other suggestions—how to name and structure Java files. However, guidelines commonly fail to significantly impact software quality, likely because they lack the motivation for their relevance [17]. For example, the aforementioned suggestions are justified because "[c]ode conventions improve the readability of the software" [16] without any empirical evidence of that claim. Furthermore, guideline conformance is difficult to assess and hence seldom done in practice [15]. The latter shortcoming was addressed by introducing metrics-based approaches where metrics were devised to measure relevant attributes of software artifacts. Among others, *lines of code* [18] and *cyclomatic complexity* [19] were used to evaluate software quality automatically. Nevertheless, most metrics continue to lack justification of their relevance [14, 20–22].

Quality models To overcome the relevance shortcoming, quality models aggregated metrics into hierarchical trees of criteria [23, 24]. The leaf nodes are specific enough to be operationalized as an evaluation metric, while the aggregation into higher-level quality characteristics provided the justification for their relevance. For example, low-level concepts such as *structuredness* and *conciseness* of code were justified by their aggregation to *understandability* and *maintainability*, which were widely accepted as relevant software quality characteristics [24]. However, hierarchical models suffered from unclear decomposition rules and constrained levels of granularity, which were either too abstract to be operationalized or too detailed, disconnecting the applicable metrics from their rationale [14, 15].

Quality meta-models The popularity of quality models necessitated a structure for the proposed models [25]. Meta-models like the Goal Question Metric approach by Basili et al. [26] and the factor-strategy quality meta-model by Marinescu and Ratiu [27] provide this overarching structure. Deissenboeck et al. [28] contribute the DAP classification for quality models, which categorizes the aim of a quality model to be to *define* (D), *assess* (A), or *predict* (P). The publication further postulates quality meta-models as the "model of the constructs and rules needed to build specific quality models." [28].

Activity-based quality models In addition to the shortcomings that existing quality models continued to suffer, the elements populating these models were found to be heterogeneous [15]—i.e., properties of a *system* were mixed with properties of *activities in which the system is used*. For example, the maintainability branch in the software quality characteristics tree by Boehm et al. [29] contains both system properties like the *structuredness* of a software artifact, but also attributes of activities in which these artifacts are used, like *modifiability*. The latter describes the *activity* of

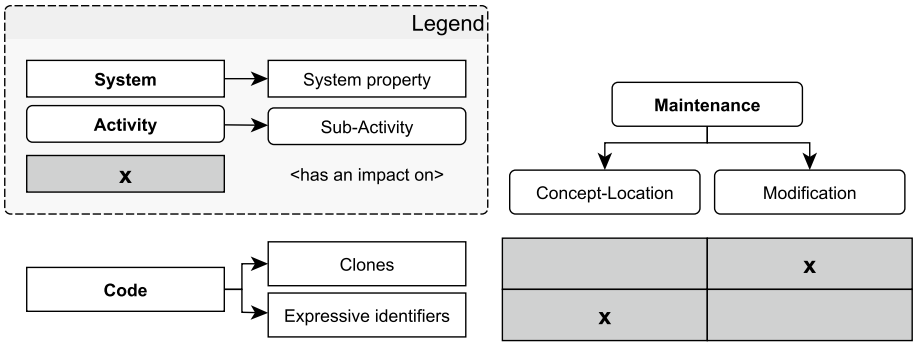

Fig. 1 Excerpt from the activity-based quality model for maintainability

modifying an artifact rather than a system property, despite the adjective's nominalization suggesting otherwise.

So far, no clear rule for distinguishing a system from an activity property has been proposed. We derived two heuristics from the implicit argumentation of previous publications [15]. First, if a property involves an additional agent (e.g., *testability* involves a *test engineer*, *modifiability* involves a *modifier*, although not necessarily human), then it represents how the system is used—i.e., an activity property. The second heuristic comes in the form of a syntactical criterion:

- Nominalized adjectives (e.g., structured-ness, conciseness) tend to be **system properties**
- Nominalized verbs (e.g., modify-ability, access-ability, augment-ability) tend to be **activity properties**

Interpreting activity properties as system properties ignores an underlying impact relationship. For example, interpreting *modifiability* as the *system* property of how receptive it is to change omits that actual system properties (e.g., whether the system is digital or analog or who has writing access rights) *impact* the ability of a stakeholder to modify the system, which is an activity property.

To address the issue of heterogeneous properties, Deissenboeck et al. introduced *activity-based quality models* [14, 15], which separate system properties from activity properties and form two distinct, orthogonal dimensions. The model expresses quality as the impact of system properties on activity properties. Figure 1 visualizes a simplified version of the quality model [15], showing how code clones impact the modification sub-activity and expressive identifiers impact the concept-location sub-activity.

The activity-based quality model was successfully applied to usability [30], security [31], and service-oriented architecture [32] before Wagner et al. distilled a comprehensive activity-based meta-model in the scope of the Quamoco project [33, 34]. In parallel, the original use case of the activity-based quality model, which focused on maintainability, received extensive tool support [35, 36] contributing evidence to the operationalization of quality models in practice [37].

Activity-based quality models solve limitations of previous quality models at the cost of increased complexity, which manifests in additional challenges to operationalize and communicate the notion of quality [38]. However, the complexity of these models is necessary to tackle the faceted concept of quality [38, 39]. Research continuously tackles the inability of activity-based quality models to assess artifact quality and distinguish quality levels [40]. For example, weights empirically derived from historical data replaced expert-based propositions [41], and Bayesian networks were utilized to model the impact relationships [42].

2.2 Mapping to requirements quality research

In the following, we draw a parallel of the evolution of quality research between the areas of software engineering and requirements engineering.

Metrics and quality models Similar to software quality, requirements quality research historically originated from proposing metrics like *passive voice* of requirements sentences [43] or *sentence length* [44], which are associated with bad quality of requirements specifications. An ongoing research endeavor [7] collects these quality factors and indicats their limitations. Most existing publications either fail to gauge the impact of these metrics [45] or explicitly disregard their relationship [46]. Requirements quality models [47, 48] integrate these factors into larger frameworks but often remain vague on their notion of impact.

The investigation of impact is often limited to a comparison between the quality factor and practitioners' subjective, general perception of the quality of the requirements entities [49]. Wilson et al. contribute a first impact matrix between quality indicators and quality attributes [50], but the latter suffers from the same system and activity properties heterogeneity. Similarly, Yang et al. [51] state that "[a]mbiguity is therefore not a property just of a text, but a conjoint property of the text and of the interpretations held by a group

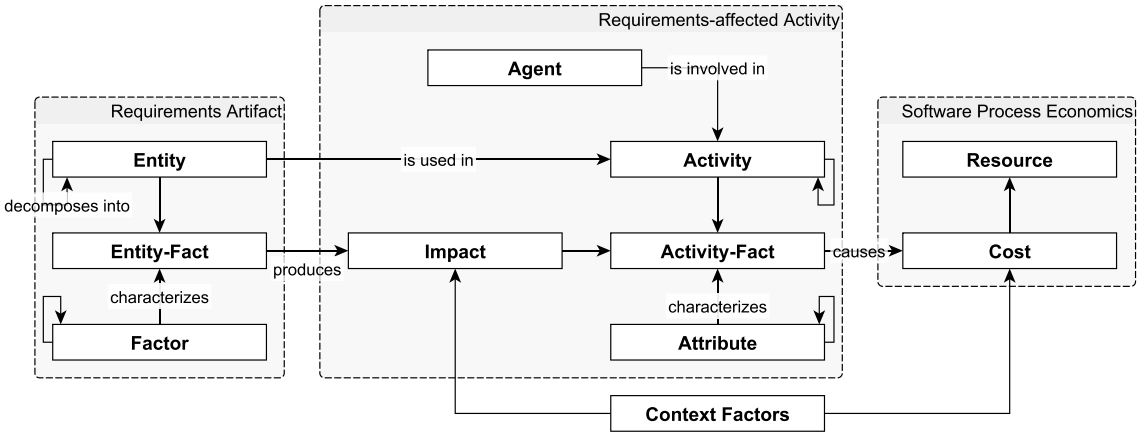

Fig. 2 Concepts of the requirements quality theory

of readers of that text," exposing the necessary distinction between system and activity properties.

Activity-based requirements quality A large portion of requirements quality research exhibits the same shortcomings identified and overcome by software quality research, namely that (1) requirements quality factors lack relevance due to their unknown impact, which in turn inhibits adoption in practice, and (2) the terminology of requirements quality aspects confuses system and activity properties.

Femmer et al. apply the activity-based quality perspective to requirements engineering by proposing the activity-based requirements engineering quality model (ABRE-QM) [52]. This model leverages the insights from activity-based software quality models [15, 17, 33] and shows that the quality of requirements depends on the impact they have on the activities in which they are used. However, despite the authors' call for action [53], ABRE-QM saw little adoption in research as demonstrated in recent systematic investigations of the requirements quality literature [6, 7].

The ABRE-QM example above raises the concern that requirements quality researchers do not properly utilize the activity-based approach successfully employed in software quality research. In this manuscript, we want to encourage further research on this approach by presenting a revised requirements quality theory, a thorough investigation of the requirements quality literature verifying the hypotheses from previous studies [6, 7], and a consequent research roadmap.

3 Requirements quality theory

We generated a harmonized requirements quality theory (RQT) by consolidating the evolution of software quality models described in Sect. 2.1, their application in requirements engineering as described in Sect. 2.2, and alignment to the established Quamoco quality model [34]. In terms of theory types [54], the RQT is both *explanatory*, as it explains the notion of requirements quality, and *prescriptive*, as it prescribes how to report contributions to requirements quality. The building blocks of the theory are described in Sect. 3.1 and illustrated with an example in Sect. 3.2.

3.1 Theory

The concepts that constitute this theory are visualized in Fig. 2, and each concept is described in Table 1. The model represents an evolution of the original activity-based requirements engineering quality model (ABRE-QM) proposed by Femmer et al. [52]. Here, we present changes to the original model.

The artifact-related section of the model (left part of Fig. 2) is largely equivalent to the original publications [15, 52]. Entities represent requirements artifacts of different granularity [5], which can be decomposed into further entities. For example, a requirements specification can be decomposed into sections, which in turn consist of paragraphs and sentences or requirements. We consider an artifact to be a high-level requirements entity and hence do not explicitly add the *artifact* to the model, deviating from the original [52]. Similarly, factors can be decomposed into sub-factors to accommodate composite factors. For example, Antinyan et al. [58] position their proposed quality factor of *conjunctive complexity* as a sub-factor of *syntactical complexity*.

The activity-related section of the model (middle part of Fig. 2) again adapts the original models [15, 52]. The concept *activity* does not represent common requirements activities, like elicitation, analysis, and validation [59], but rather every process that takes a requirements entity as input

Table 1 Explanation and origin of theory concepts

Concept	Explanation	Origin
Entity	A requirements artifact or part thereof	[52]
Factor	"[A] normative metric which maps a textual requirement of a specific granularity" [7] to a numerical output	[15, 52]
Entity-fact	A composition of one entity and one factor	[15]
Agent	Any person, group of people, or automatism involved in an activity	[52]
Activity	An activity in which the entity is used	[15]
Attribute	A measurable property of an activity	[30]
Activity-fact	A composition of one activity and one attribute	
Impact	The impact of a fact on an activity-fact	[15, 52]
Context factor	A factor describing the context of the impact relationship	[55, 56]
Cost	The magnitude of cost associated with an activity-fact	[56]
Resource	The resource affected by the economical impact	[56, 57]

and produces an output. This includes some requirements activities (like analysis and validation, which use requirements as input) but not others (like elicitation, which often does not presuppose existing requirements). Hence, we rather refer to them as *requirements-affected activities*. These further include implicit sub-activities (e.g., *understanding* and *interpreting* an entity), which can be aggregated with other, more explicit sub-activities (e.g., *test case design*) to form high-level activities (e.g., *validation*). The decomposition relationship of the activity concept accommodates this aggregation. To accommodate not only human actors involved in activities but also any automatism like requirements processing tools [60] we abstract the concept of *stakeholder* to *agent*.

We generalized the impact concept in this theory. While previous models assumed that impact is categorical (i.e., the occurrence of a fact has either a positive, negative, or no impact at all, like in Fig. 1 [15] or linear (i.e., the larger the evaluation of a quality factor, the better/worse is its quality), we consider the impact to model any kind of relationship between Entity-facts and Activity-facts. This opens up the theory to more complex relationships, which can model the actual impact more accurately and allows to compare the impact of quality factors with each other.

Two concepts were added to the model. First, the impact was related to an *Activity-fact* composed of an activity and an attribute as proposed by Winter et al. [30]. This way, the structure of the variables on the two sides of the impact relationship is mirrored. Furthermore, the necessity to associate an impact with a measurable property of an activity is emphasized. Second, context factors also influence the impact of an Entity-fact on an Activity-fact. As recognized by previous publications [55, 56], the impact differs depending on external factors related to, among others, the organization and the people involved [61].

The economic section of the model (right part of Fig. 2) is a novel addition to previous iterations of the activity-based models [15, 34, 52]. As long as the subsequent *economic* impact of an Activity-fact is unknown, the Entity-fact that produces the Impact on this Activity-fact will remain neglected [56, 57]. Hence, the software process economics perspective introduces a *Cost* for a specific *Resource* such as time or money.

3.2 Example

In this section, we illustrate the RQT with a fictitious example to demonstrate its application. The example is additionally visualized in Fig. 3.

In this example, a customer's requirements were elicited and documented in a requirements specification containing the entity *user story 42*. One relevant quality factor used by the organization responsible for implementing the requirements is template *conformance*, which prescribes that all user stories must follow the Connextra template [62] "As a ⟨role⟩ I want to ⟨goal⟩ so that ⟨benefit⟩." This quality factor maps the entity to a categorical value, containing—among others—the values *conform*, *missing role*, and *missing all elements*. In this example, the role is omitted from the user story. Hence, the quality factor template conformance is evaluated to *missing role*, which constitutes the entity-fact (yellow box in Fig. 3).

The organization uses this user story in a subsequent, requirements-affected *development* activity, where a different stakeholder—the developer—is responsible for translating the entity into code. This activity can be decomposed into two distinct sub-activities: *understanding* the entity and *programming* the respective implementation.

One desired attribute of the activity understanding is *determinism*—i.e., a requirements entity should have only one unique interpretation. Possible variations of the interpretation and, therefore, the subsequent translation of a requirement must be avoided. Because the *conformance* quality factor is evaluated to *missing role* on the *user story*

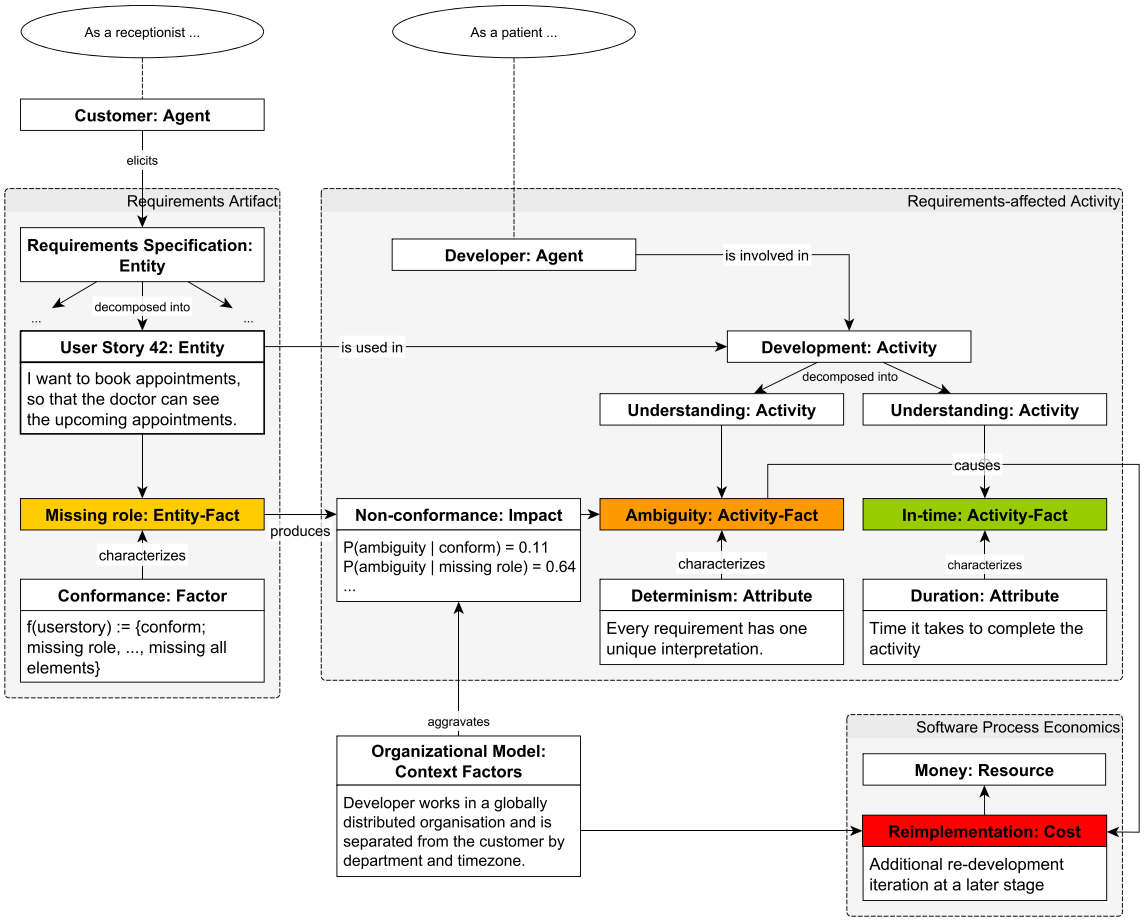

Fig. 3 Exemplary instantiation of the theory

entity, the *understanding* activity is less *deterministic*, as the developer can make a different assumption about the role implied by the requirement. The understanding activity has become ambiguous, which constitutes the *activity-fact* (orange box in Fig. 3).

The relationship between the entity-fact and the activity-fact is the *impact* of the quality factor. Instead of limiting the impact concept to categorical values (e.g., either *has an impact* or *has no impact*), the RQT enables more complex impact relationships. In this fictitious example, the quality factor value *missing role* is associated with a 64% chance of making the understanding sub-activity ambiguous. This relationship can be determined empirically via experimental research investigating the likelihood of the different values of the conformance quality factor reducing the determinism of the understanding sub-activity.

The programming sub-activity may go unaffected by the entity-fact that the conformance has a value of *missing role* (green box in Fig. 3): regardless of the agent's interpretation of the requirements entity, the programming sub-activity will remain unaffected in respect to the relevant attribute *duration* under the assumption of a similar user interface for both roles. Whether the feature is coded for the role receptionist (as the customer intended) or patient (as the developer assumed) does not significantly change the duration of the sub-activity if the user interfaces only barely differ.

The significant impact on understanding is influenced by the organizational model, which is one relevant *context factor*. Since the organization is globally distributed and the two involved agents are unlikely to have informal interactions, the impact is amplified. In contrast, in a small organization where all involved agents share an office, the impact can be alleviated as missing information is recovered through informal communication. Similarly, the software development process model may significantly influence the impact of the quality factor, and the use of an agile approach may reduce the impact by encouraging communication between the customer and developer. The context factors significantly influence the impact and, therefore, have to be included in the relationship between entity-facts and activity-facts.

The reduced determinism of the understanding activity has an economic effect—i.e., the less deterministic the activity is, the more the implementation needs to be revised, which costs money and time (red box in Fig. 3). Context

factors influence the extent of this effect as, for example, a re-implementation can be more costly in larger organizations due to organizational overhead.

For the sake of brevity, the example omits the following aspects: (1) the example limits the number of elements populating the relationship. More quality factors of the entity, activities, attributes of activities, and context factors are possibly involved in the relationship. (2) Interaction effects between quality factors and context factors are plausible but not reported here.

However, the example demonstrates how adherence to this activity-based RQT elevates requirements quality factors from normative rules (i.e., user stories must conform the template for the sake of it) to empirically backed impact predictions (i.e., user stories must conform the template to mitigate ambiguous interpretations and avoid implementation cost).

4 State of research

Despite the publication of the ABRE-QM [52] and its authors' proposition to adapt the quality meta-model for future requirements quality research [53], recent systematic reviews raised concerns regarding a perspective on requirements quality limited to the artifact-related section of the model (left part of Fig. 2) [6, 7].

To validate these concerns, we formulate the following research question. **How are the concepts of the requirements quality theory reported in requirements quality literature?** Answering this research question requires extracting information from a population of publications; accordingly, we employ survey research as our approach to gain insight into the current state of research. We follow the survey guidelines by Molléri et al. [63] and report our survey in the following subsections. All supplementary material for replicating this study is available in our replication package.[1]

4.1 Survey objects

The target population of our survey is the requirements quality literature dealing with quality factors in requirements artifacts. In a previous research endeavor [7], we conducted a systematic study on requirements quality factors, including a sample of 57 primary studies. To our knowledge, this is the only sample that fulfills our aforementioned requirements. This classifies the sampling as non-probabilistic, more specifically convenience sampling [63].

4.2 Study design

We follow the recommended practices for the survey research process and report our steps accordingly [63]. However, we disregarded steps that only apply to surveys with human subjects, such as *participant recruitment* and *response management*.

We derived the *definition of the research objectives* in the form of the research question directly from previous research [6, 7, 53]. We established a *study plan*, rigorously documenting all research progress and justifications for any deviations during the process. We *identified and characterized the population* of our survey and executed our *sampling plan* as described in Sect. 4.1.

For our *instrument design*, we maintained two artifacts. We created an extraction guideline based on the RQT concepts. Each concept of the RQT was associated with one or more categorical variables, each containing a set of codes that represented *if* and *how* the concept was reported. The codes were created ad hoc in the first iteration of extraction and refined based on discussions and theoretical background in the second iteration.

The extent of the codes varied. The codes that represent how the concept *entity* is reported are, for example, *explicit* and *implicit*. An entity is either reported explicitly if its scope and form are clear. It is reported implicitly if the authors just report that the factor applies to a "requirement" without defining whether this is a single or multiple natural language sentence, whether the language is constrained or not, or whether it assumes a full sentence at all.

The codes of other concepts were more complex and grouped into distinct categories. For example, the codes of the concept *Factor* were split into two groups, representing both the *explicitness* when reporting a factor (i.e., whether the factor is explicitly *reported* or *referenced* from another publication) and the *form* in which the factor is reported (i.e., if the factor is represented with a *textual description* or defined using a logical or mathematical *formula*).

The first author extracted the appropriate code for each concept in the requirements quality theory from each publication. The extractions for each publication in the sample were recorded in a spreadsheet. For *instrument validation*, the second author of this manuscript independently performed the extraction task using the guideline on six ($\approx 10\%$) publications randomly sampled from the survey objects. The second author performed the extraction on two of these six publications as training, and the remaining four were used to calculate the inter-rater reliability between the first and second author.

The task overlap achieved an percentage agreement [64] of 83.3%, whereas Cohen's Kappa yields a *moderate*

[1] Available at https://doi.org/10.5281/zenodo.8167598.

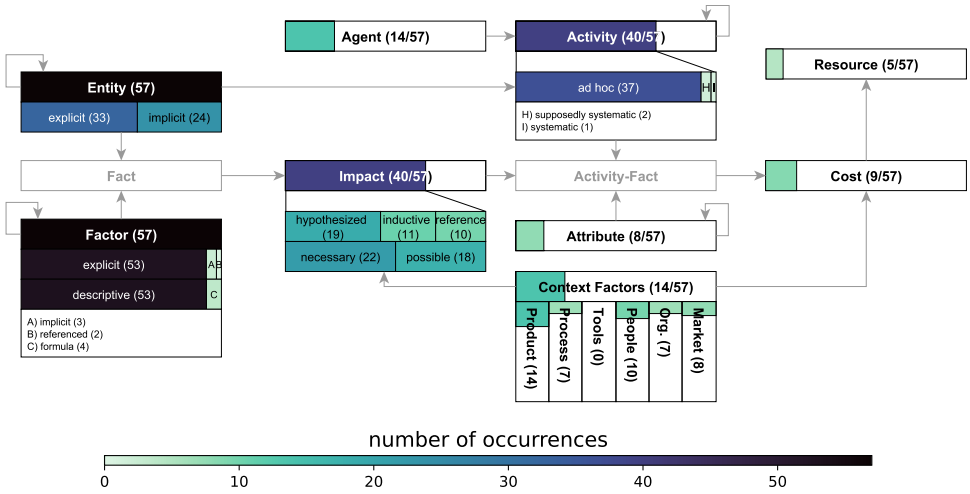

Fig. 4 Survey results depicting the distribution of codes

agreement of 54.2%. As Cohen's Kappa is unreliable for uneven marginal distributions [65], we calculated the more robust S-Score [66]—yielding a *good* agreement of 76.8% —which we deem sufficient for assessing the inter-rater reliability.

We used the codes in the *data analysis* phase to generate descriptive statistics on which we based our interpretation of the state of requirements quality. These form a quantified foundation for interpreting the state of requirements quality literature with respect to the research question. For final *reporting*, we adapted established reporting guidelines [63] and disclosed all material in a reusable replication package.

4.3 Study results

Figure 4 visualizes the distribution of the relevant codes among all concepts included in the requirements quality theory. Each concept is overlaid with a bar representing how many of the 57 publications contained the concept. The row below each concept represents its dimensions derived from the appropriate codes.

Though both entities and factors are explicitly reported in all 57 publications of the sample, a large portion ($24/57 = 42.1\%$) of entities is reported implicitly—i.e., the entity's scope is not clear. This occurs mostly because authors attach the reported quality factor to the entity *requirement* without specifying the scope or form of the entity. Montgomery et al. [6] have already noted this shortcoming in the requirements quality literature and it represents a terminological ambiguity in the research domain.

Seventeen out of 57 publications (29.8%) do not report any impact on activities (code *N/A*) and hence neglect the practical relevance of the proposed quality factors. Agents are only reported in 14 (24.6%) of all publications. Activities are—when reported—predominantly elicited *ad hoc* ($37/40 = 92\%$) and rarely *systematically*—i.e., when activities impacted by a quality factor are discussed, the identification of activities has no systematic approach. Attributes are also only rarely reported ($8/57 = 14\%$).

We grouped the codes classifying how *impact* is reported into four distinct dimensions, two of which are reported here. The *evidence* for the impact—when at all reported—is dominantly hypothesized ($19/40 = 47.5\%$) and rarely either inductive ($11/40 = 27.5\%$) or referenced ($10/40 = 25\%$), i.e., draws the evidence from another publication. Previous studies [6, 7] have also noted this dominance of anecdotal, non-empirical evidence. The *modality* of impact relationships is balanced between *necessary* and *possible*—i.e., the impact of quality factors is reported almost equally often to be certain or potential. The remaining two dimensions of impact (*generality* and *frame of reference*) yielded no additional insight into the surveyed objects and are hence not reported here but contained in the replication package.

Context factors are almost completely neglected and only reported to a degree varying between zero (no publication reports the influence of any *tools*) and 24.6% (14 out of 57 publications reporting *product*-related factors, e.g., the system's size or type).

Both *cost* and *resources* are reported only rarely ($9/57 = 15.8\%$ and $5/57 = 8.8\%$, respectively) and, if so, only hypothesized or referenced, never determined empirically. Money and time are mentioned as the resources affected by activity impact, and the cost is only estimated in terms of expected change (e.g., "*reduction* of the time spent" [46]) or general magnitude (e.g., "*significant* amounts of money" [67]).

4.4 Interpretation

In this section, we interpret the results presented in Sect. 4.3 and answer the research question.

Publications in the requirements quality literature adhere to the RQT to a varying degree. While all publications in the sample mentioned both an entity and a quality factor, activity-related concepts, context factors, and the economic impact are often neglected. Failing to consider the context factors severely threatens the external validity of the proposed quality factors [55, 56] and neglecting the economic impact risks undermines their acceptance [56, 57].

Context factors and economic impact are arguably more challenging to investigate [68]; however, we emphasize that the lack of activity perspective when proposing quality factors is critical for several reasons. The complete negligence of a quality factor's impact limits the factor to a normative, unmotivated prescription and challenges its practical relevance [52], which in turn promotes skepticism regarding requirements quality factors in industry [8–11].

The survey emphasized two additional shortcomings in the field of requirements quality research. First, the tendency to elicit activities *ad hoc* when discussing the impact of requirements quality factors bears the risk of missing other important impacts. Most publications discuss a hypothesized impact of a quality factor on a non-systematically selected activity or set of activities. This selection is usually justified by anecdotal or folkloric circumstances, like "[a]mbiguous requirements may bring about misinterpretations among stakeholders, and prompt a few issues" [69].

While these impact relationships are neither empirically proven nor falsified, the non-systematic selection of activities can disregard other impact relationships. Femmer et al. [52] demonstrated that a systematic elicitation of activities could reveal both positive and negative impacts by the same quality factor. For example, the factor *free of UI design details*, which states that an "artifact should describe the problem domain instead of the solution domain" [52], will positively affect maintainability, as UI details are volatile in the beginning and require a lot of change management if specified in a requirement. Conversely, the same factor negatively impacts understandability, as the presence of UI design makes requirements more comprehensible.

Second, while activities are not reported consistently, attributes of activities are reported even less. Attributes represent measurable characteristics of activities; for example, the activity *understanding* can be quantified by its attribute *level of agreement* [58, 70] or a *readability index* [71]. Neglecting the quantifiable attributes of activities impedes an empirical evaluation of a quality factor impact because it omits the measurement instrument for the dependent variable (i.e., the activity-fact) in the impact relationship [30].

We conclude that the requirements quality theory is implicitly embedded in the requirements quality literature. However, insufficient adherence to it results in several limitations when reporting new requirements quality factors. While the artifact-centric theory concepts are commonly covered, activity-centric concepts, context factors, and economic concepts receive less attention, which decreases these publications' practical relevance. With this study, we empirically confirm the concerns voiced in previous investigations of the requirements quality literature [6, 7].

4.5 Threats to validity of this research

We discuss the threats to validity proposed by Wohlin et al. [72] and extended by Molléri et al. [63].

Internal validity We acknowledge a threat to internal validity due to sampling of publications. The method of object selection [6, 7] is deemed sufficiently rigorous to derive an initial theory.

Construct validity The constructs in this study—i.e., the elements of the theory—are established strictly following mature quality theories from the field of software quality. This ensures the alignment between the underlying theory and measurement constructs.

The lack of a theory to which the surveyed publications could have adhered when reporting quality factors resulted in the concepts of requirements quality often being embedded implicitly, complicating the extraction task. We minimized the resulting threat to internal validity through independent labeling and calculating appropriate inter-rater reliability metrics [65].

External validity The selected sample of publications [7] is constrained to empirical contributions to requirements quality research [6]. This limits the conclusion validity of the type of evidence for the *impact* concept, as non-empirical work could contribute *theoretical* evidence for impact relationships. For example, the impact of quality factors like *nominalization* [73] can be derived deductively by referring to valency reduction caused by nominalization [74]. While publications utilizing linguistic theory are unknown to the authors, a valid conclusion regarding this type of evidence requires a more thorough extension of the sampling strategy.

5 Research roadmap

Femmer et al. proposed an initial research roadmap detailing how to advance the field of requirements quality research [53]. Based on concerns of previous studies [6, 7] and the survey of the state of research reported in this study, we assess and update the three suggested steps by Femmer et al. [53]:

1. Creation of "a reference artifact and a usage model" eliciting typical entities, activities, and agents.
2. Creation of "a taxonomy of quality factors" as a central, accessible repository of quality factors.
3. Creation of "a taxonomy of impacts" as a catalog of impacts from quality factors onto activities.

We reflect on these proposed research streams in Sects. 5.1 to 5.3 and add three further proposals in Sects. 5.4 to 5.6. Because these research streams are grounded in the experiences from the software quality research, we expect contributions to them to promote requirements quality research that is relevant to practice.

5.1 Artifact and usage model

Mendez et al. have contributed a reference artifact model for requirements engineering [5, 75] based on their fundamental positioning on artifact orientation [76, 77]. The AMDiRE approach constitutes a domain-agnostic reference for artifact types and serves the purpose requested by Femmer et al. [53] in that it can be tailored toward any industry context to model an artifact structure.

While the elicitation of human [78] and non-human, automatic agents [79] has been addressed, a reference model for activities requires explicit attention in the literature. More importantly, with the update of the requirements quality theory over the initial ABRE-QM [52], we argue that a reference model for requirements-affected activities needs to provide *attributes* to quantify each activity. Such attributes enable an empirical assessment of the impact of quality factors.

Additionally, a majority of publications reporting an impacted activity mention some variation of *understanding* or *interpreting* ($32/40 = 80\%$). We assume that every requirements-affected activity comprises an initial *interpretation* sub-activity. However, such composition is obscured by the lack of a proper reference model for requirements-affected activities accounting for their aggregated nature.

It is conceivable that the *interpretation* sub-activity is most prone to defects, which explains the research community's focus on *ambiguity* [6], as ambiguity represents the non-determinism of an interpretation. We argue that a proper reference model for requirements-affected activities accounting for their aggregated nature can steer research toward identifying critical sub-activities—i.e., the ones most prone to impacting subsequent activities.

5.2 Taxonomy of quality factors

Requirements quality factors [7, 53] are the cornerstone of artifact-centric quality assurance. The requirements quality factor ontology [7] furthered this research stream. Although the ontology is in an early stage and requires additional iterations, quality factors and related objects—such as data sets and automation approaches—are now collected in a central repository.

5.3 Taxonomy of impacts

The taxonomy of impacts that Femmer et al. [53] deem the necessary final step of the roadmap has to be extended. Previous quality models—including the ABRE-QM [52]—consider only categorical or, at most, linear impact relationships. Therefore, a taxonomy seemed sufficient to record "a list of well-examined effects of quality factors on activities" [53]. We argue that the impact relationship can be more complex and requires a more general representation—i.e., rather than aiming for a taxonomy of impacts, we argue for developing an *impact framework*.

Given the evaluation of quality factors on requirements entities on one side and the evaluation of activity attributes on the other side, the impact relationship between these variables can be formulated as a regression problem. Instead of relying on experts to hypothesize the (categorical) type or (linear) extent of an impact, more complex relationships can be determined using, for example, Bayesian data analysis [80]. Consequently, this research stream aims to develop an impact framework capable of determining these impact relationships based on statistical instruments given sufficient data.

5.4 Context factors

Context factors must be considered in the impact relationship to operationalize the requirements quality theory [55]. Large-scale endeavors acknowledge the importance of context factors in regard to requirements quality [1], yet no unified collection of context factors relevant to requirements engineering exists. Established sets of software engineering context factors [61, 81] can be used as a starting point but require a dedicated investigation from the requirements engineering perspective.

A clear set of relevant context factors can support developing reporting guidelines for empirical studies on requirements quality and enable context-driven research [82]. While empirical software and requirements engineering publications typically strive for generalizability [81], scoping an empirical study according to the given context factors allows the data collected in that study to be integrated into the impact framework as outlined in Sect. 5.3. Conversely, reporting the limited scope of a study enables a general requirements quality theory that can be assembled from multiple studies in well-defined contexts.

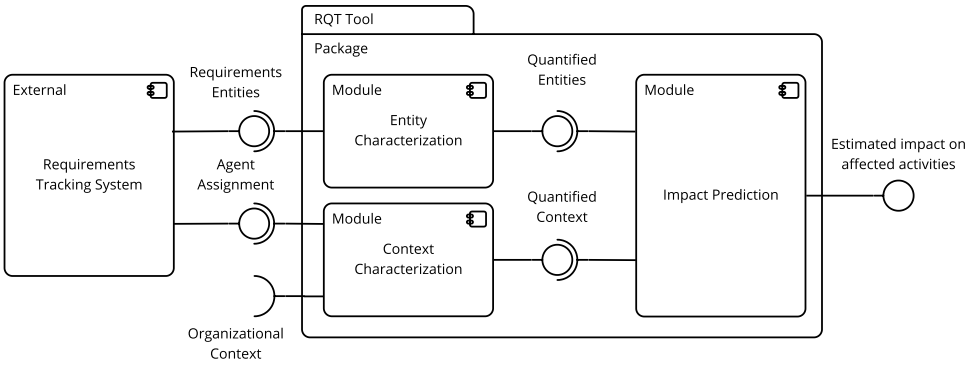

Fig. 5 Architectural overview of the proposed tool-support

5.5 Economic impact

With the addition of economic concepts in the requirements quality theory, a research stream should be dedicated to the economic impact of activity facts. The impact relationship between quality factors and activities already benefits the acceptance of those factors for quality assurance in practice [53]. Adding an economic perspective—i.e., what amount of which resource a change of a certain activity-fact entails—can further bridge the gap between the normative, artifact-centric quality factors on one side and an economic decision-making process on the other side [57]. Since the purpose of quality factors is to support quality assurance in industry, understanding this economic perspective is of high priority despite the complexity of the topic.

5.6 Tool support

We aim to make the RQT applicable to the industrial context through the development of tool support. The components necessary to realize this tool support are visualized in Fig. 5. The goal of the tool is to estimate the impact of requirements entities and their context on the attributes of requirements-affected activities.

To this end, the tool needs an interface to the requirements entities, context information about the involved agents, and context information about the organization. The former two are often available in a requirements tracking system like Jira, while the latter a company likely has to generate and provide manually [83].[2]

Once provided with the necessary information, the tool characterizes both entities and context, i.e., quantifies the natural language requirements entities and the elusive factors determining the context. The quantified entities and context serve as input to the impact prediction model as described in Sect. 5.3, estimating the impact on the attributes of the requirements-affected activities, which in turn enables quantifying the economic impact as described in Sect. 5.5.

The realization of this tool depends on the previously described streams of research to identify valid quality factors (Sect. 5.2), context factors (Sect. 5.4), and activity attributes (Sect. 5.1). For the tool to provide an automated impact prediction the following automation modules must be realized:

1. Automatic entity characterization: a shared architecture to automatically evaluate the requirements quality factors collected in the quality factor ontology [7]
2. Automatic impact prediction: an accessible statistical model estimating the impact of quantified entities and context on affected activities, trained on historical data.

Developing this tool while adhering to open science principles will allow scholars to propose new quality and context factors, customize relevant activity attributes, and contribute historical data to improve the impact estimation of the prediction model. We invite contributions to the implementation and maintenance of the tool via its dedicated repository on Github.[3]

6 Conclusion

In this manuscript, we investigated the software quality literature and the application of the activity-based quality perspective to the requirements engineering domain. We extend the work of Femmer et al. [52] by proposing an evolved and harmonized requirements quality theory and assess the adherence of the requirements quality literature to this theory. Our survey confirms the bias toward artifact-centric and the negligence of activity-centric concepts, which was noted in previous secondary studies [6, 7]. Finally, we update the requirements quality research roadmap initiated by Femmer

[2] https://www.atlassian.com/software/jira.

[3] Available at https://github.com/JulianFrattini/rqt-tool. An archived version is accessible at https://doi.org/10.5281/zenodo.8167541.

et al. [53] to guide future contributions in the requirements quality research domain.

We are confident that the harmonized requirements quality theory provides the necessary guidance to propel requirements quality research and establish a common understanding of quality that is operationalizable in practice. We invite fellow researchers to contribute to the theory and the requirements quality research field in adherence to it.

Acknowledgements This work was supported by the KKS foundation through the S.E.R.T. Research Profile project at Blekinge Institute of Technology. We further thank our colleagues and the reviewers for their constructive feedback which strengthened this article.

Funding Open access funding provided by Blekinge Institute of Technology.

References

1. Méndez Fernández D, Wagner S, Kalinowski M, Felderer M, Mafra P, Vetrò A, Conte T, Christiansson M-T, Greer D, Lassenius C et al (2017) Naming the pain in requirements engineering: contemporary problems, causes, and effects in practice. Empir Softw Eng 22(5):2298–2338
2. Wagner S, Méndez Fernández D, Felderer M, Vetrò A, Kalinowski M, Wieringa R, Pfahl D, Conte T, Christiansson M-T, Greer D et al (2019) Status quo in requirements engineering: a theory and a global family of surveys. ACM Trans Softw Eng Methodol (TOSEM) 28(2):1–48
3. Damian D, Chisan J (2006) An empirical study of the complex relationships between requirements engineering processes and other processes that lead to payoffs in productivity, quality, and risk management. IEEE Trans Softw Eng 32(7):433–453
4. Boehm BW, Papaccio PN (1988) Understanding and controlling software costs. IEEE Trans Softw Eng 14(10):1462–1477
5. Méndez Fernández D, Böhm W, Vogelsang A, Mund J, Broy M, Kuhrmann M, Weyer T (2019) Artefacts in software engineering: a fundamental positioning. Softw Syst Model 18(5):2777–2786
6. Montgomery L, Fucci D, Bouraffa A, Scholz L, Maalej W (2021) Empirical research on requirements quality: a systematic mapping study. Requir Eng 27:183–209
7. Frattini J, Montgomery L, Fischbach J, Unterkalmsteiner M, Mendez D, Fucci D (2022) A live extensible ontology of quality factors for textual requirements. arXiv preprint arXiv:2206.05959
8. Franch X, Mendez D, Vogelsang A, Heldal R, Knauss E, Oriol M, Travassos G, Carver JC, Zimmermann T (2020) How do practitioners perceive the relevance of requirements engineering research? IEEE Trans Softw Eng48(6):1947–1964
9. Berry D, Gacitua R, Sawyer P, Tjong SF (2012) The case for dumb requirements engineering tools. In: International working conference on requirements engineering: foundation for software quality. Springer, pp 211–217
10. Femmer H (2018) Requirements quality defect detection with the Qualicen requirements scout. In: REFSQ workshops
11. Phalp KT, Vincent J, Cox K (2007) Assessing the quality of use case descriptions. Softw Qual J 15(1):69–97
12. Lindland OI, Sindre G, Solvberg A (1994) Understanding quality in conceptual modeling. IEEE Softw 11(2):42–49
13. Pohl K (1993) The three dimensions of requirements engineering. In: International Conference on advanced information systems engineering. Springer, Berlin, pp 275–292
14. Broy M, Deißenböck F, Pizka M (2005) A holistic approach to software quality at work. In: Proceedings of the 3rd world congress for software quality (3WCSQ)
15. Deissenboeck F, Wagner S, Pizka M, Teuchert S, Girard J-F (2007) An activity-based quality model for maintainability. In: 2007 IEEE international conference on software maintenance. IEEE, pp. 184–193
16. King P, Naughton P, DeMoney M, Kanerva J, Walrath K, Hommel S (2021) Code conventions for the Java programming language. Technical report, Sun Microsystems, Inc., Mountain View, CA, USA 1999
17. Broy M, Deissenboeck F, Pizka M (2006) Demystifying maintainability. In: Proceedings of the 2006 international workshop on software quality, pp 21–26
18. Albrecht AJ, Gaffney JE (1983) Software function, source lines of code, and development effort prediction: a software science validation. IEEE Trans Softw Eng 6:639–648
19. McCabe TJ (1976) A complexity measure. IEEE Trans Softw Eng 4:308–320
20. Rosenberg J (1997) Some misconceptions about lines of code. In: Proceedings Fourth international software metrics symposium, pp. 137–142. IEEE
21. Khoshgoftaar TM, Munson JC (1990) The lines of code metric as a predictor of program faults: a critical analysis. In: Proceedings fourteenth annual international computer software and applications conference. IEEE Computer Society, pp 408–409
22. Shepperd M (1988) A critique of cyclomatic complexity as a software metric. Softw Eng J 3(2):30–36
23. McCall JA (1977) Factors in software quality. US Rome Air development center reports
24. Boehm BW, Brown JR, Lipow M (1976) Quantitative evaluation of software quality. In: proceedings of the 2nd international conference on software engineering, pp 592–605
25. Kitchenham B, Linkman S, Pasquini A, Nanni V (1997) The squid approach to defining a quality model. Softw Qual J 6(3):211–233
26. Basili VR, Caldiera G, Rombach HD (1994) The goal question metric approach. Encyclopedia Softw Eng 1:528–532
27. Marinescu R, Ratiu D (2004) Quantifying the quality of object-oriented design: the factor-strategy model. In: 11th working conference on reverse engineering. IEEE, pp 192–201
28. Deissenboeck F, Juergens E, Lochmann K, Wagner S (2009) Software quality models: Purposes, usage scenarios and requirements. In: 2009 ICSE workshop on software quality. IEEE, pp 9–14
29. Boehm BW, Brown JR, Kaspar H, Lipow M, MacLeod G (1978) Merritt: characteristics of software quality. North Holland, Amsterdam
30. Winter S, Wagner S, Deissenboeck F (2007) A comprehensive model of usability. In: IFIP international conference on engineering for human–computer interaction. Springer, Berlin, pp 106–122
31. Wagner S, Fernandez DM, Islam S, Lochmann K (2009) A security requirements approach for web systems. In: Workshop quality assessment in web (QAW 2009)

32. Goeb A, Lochmann K (2011) A software quality model for SOA. In: Proceedings of the 8th international workshop on software quality, pp 18–25
33. Wagner S, Lochmann K, Heinemann L, Kläs M, Trendowicz A, Plösch R, Seidi A, Goeb A, Streit J (2012) The Quamoco product quality modelling and assessment approach. In: 2012 34th international conference on software engineering (ICSE). IEEE, pp 1133–1142
34. Wagner S, Lochmann K, Winter S, Deissenboeck F, Juergens E, Herrmannsdoerfer M, Heinemann L, Kläs M, Trendowicz A, Heidrich J et al (2012) The quamoco quality meta-model
35. Deissenboeck F, Juergens E, Hummel B, Wagner S, y Parareda BM, Pizka M (2008) Tool support for continuous quality control. IEEE Softw 25(5):60–67
36. Deissenboeck F, Heinemann L, Herrmannsdoerfer M, Lochmann K, Wagner S (2011) The quamoco tool chain for quality modeling and assessment. In: 2011 33rd international conference on software engineering (ICSE). IEEE, pp 1007–1009
37. Steidl D, Deissenboeck F, Poehlmann M, Heinke R, Uhink-Mergenthaler B (2014) Continuous software quality control in practice. In: 2014 IEEE international conference on software maintenance and evolution. IEEE, pp 561–564
38. Lochmann K, Ramadani J, Wagner S (2013) Are comprehensive quality models necessary for evaluating software quality? In: Proceedings of the 9th international conference on predictive models in software engineering, pp 1–9
39. Garvin DA (1984) What does product quality really mean. Sloan Manag Rev 25:25–43
40. Kläs M, Lochmann K, Heinemann L (2011) Evaluating a quality model for software product assessments-a case study. In: Proceedings of SQMB 11
41. Wagner S, Goeb A, Heinemann L, Kläs M, Lampasona C, Lochmann K, Mayr A, Plösch R, Seidl A, Streit J et al (2015) Operationalised product quality models and assessment: the Quamoco approach. Inf Softw Technol 62:101–123
42. Wagner S (2010) A Bayesian network approach to assess and predict software quality using activity-based quality models. Inf Softw Technol 52(11):1230–1241
43. Femmer H, Kučera J, Vetrò A (2014) On the impact of passive voice requirements on domain modelling. In: Proceedings of the 8th ACM/IEEE international symposium on empirical software engineering and measurement, pp 1–4
44. Ferrari A, Gori G, Rosadini B, Trotta I, Bacherini S, Fantechi A, Gnesi S (2018) Detecting requirements defects with NLP patterns: an industrial experience in the railway domain. Empir Softw Eng 23(6):3684–3733
45. Habib MK, Wagner S, Graziotin D (2021) Detecting requirements smells with deep learning: Experiences, challenges and future work. In: 2021 IEEE 29th international requirements engineering conference workshops (REW). IEEE, pp 153–156
46. Femmer H, Fernández DM, Wagner S, Eder S (2017) Rapid quality assurance with requirements smells. J Syst Softw 123:190–213
47. Berry DM, Bucchiarone A, Gnesi S, Lami G, Trentanni G (2006) A new quality model for natural language requirements specifications. In: Proceedings of the international workshop on requirements engineering: foundation of software quality (REFSQ)
48. Lucassen G, Dalpiaz F, van der Werf JME, Brinkkemper S (2017) Improving user story practice with the Grimm method: a multiple case study in the software industry. In: International working conference on requirements engineering: foundation for software quality. Springer, Berlin, pp 235–252
49. Parra E, Dimou C, Llorens J, Moreno V, Fraga A (2015) A methodology for the classification of quality of requirements using machine learning techniques. Inf Softw Technol 67:180–195
50. Wilson WM, Rosenberg LH, Hyatt LE (1997) Automated analysis of requirement specifications. In: Proceedings of the 19th international conference on software engineering, pp 161–171
51. Yang H, De Roeck A, Gervasi V, Willis A, Nuseibeh B (2011) Analysing anaphoric ambiguity in natural language requirements. Requir Eng 16(3):163–189
52. Femmer H, Mund J, Fernández DM (2015) It's the activities, stupid! A new perspective on re quality. In: 2015 IEEE/ACM 2nd international workshop on requirements engineering and testing. IEEE, pp 13–19
53. Femmer H, Vogelsang A (2018) Requirements quality is quality in use. IEEE Softw 36(3):83–91
54. Gregor S (2006) The nature of theory in information systems. MIS Q 30:611–642
55. Mund J, Fernandez DM, Femmer H, Eckhardt J (2015) Does quality of requirements specifications matter? Combined results of two empirical studies. In: 2015 ACM/IEEE international symposium on empirical software engineering and measurement (ESEM), pp. 1–10. IEEE
56. Juergens E, Deissenboeck F (2010) How much is a clone. In: Proceedings of the 4th international workshop on software quality and maintainability, pp 79–88
57. Deissenboeck F, Pizka M (2007) The economic impact of software process variations. In: International conference on software process. Springer, Berlin, pp 259–271
58. Antinyan V, Staron M, Sandberg A, Hansson J (2016) A complexity measure for textual requirements. In: 2016 joint conference of the international workshop on software measurement and the international conference on software process and product measurement (IWSM-MENSURA). IEEE, pp 148–158
59. Sommerville I (2005) Integrated requirements engineering: a tutorial. IEEE Softw 22(1):16–23
60. Fischbach J, Frattini J, Vogelsang A, Mendez D, Unterkalmsteiner M, Wehrle A, Henao PR, Yousefi P, Juricic T, Radduenz J et al (2023) Automatic creation of acceptance tests by extracting conditionals from requirements: NLP approach and case study. J Syst Softw 197:111549
61. Petersen K, Wohlin C (2009) Context in industrial software engineering research. In: 2009 3rd international symposium on empirical software engineering and measurement. IEEE, pp 401–404
62. Cohn M (2004) User stories applied: for agile software development. Addison-Wesley Professional, Boston
63. Molléri JS, Petersen K, Mendes E (2020) An empirically evaluated checklist for surveys in software engineering. Inf Softw Technol 119:106240
64. Holsti OR (1969) Content analysis for the social sciences and humanities. (Content analysis). Addison-Wesley, Reading, MA
65. Feng GC (2015) Mistakes and how to avoid mistakes in using intercoder reliability indices. Methodol: Eur J Res Methods Behav Soc Sci 11(1):13
66. Bennett EM, Alpert R, Goldstein A (1954) Communications through limited-response questioning. Public Opin Q 18(3):303–308
67. Femmer H, Unterkalmsteiner M, Gorschek T (2017) Which requirements artifact quality defects are automatically detectable? A case study. In: 2017 IEEE 25th international requirements engineering conference workshops (REW). IEEE, pp 400–406
68. Kamata MI, Tamai T (2007) How does requirements quality relate to project success or failure? In: 15th IEEE international requirements engineering conference (RE 2007). IEEE, pp 69–78
69. Sinpang JS, Sulaiman S, Idris N (2017) Detecting ambiguity in requirements analysis using Mamdani fuzzy inference. J Telecommun Electron Comput Eng (JTEC) 9(3–4):157–162
70. Chantree F, Nuseibeh B, De Roeck A, Willis A (2006) Identifying nocuous ambiguities in natural language requirements. In: 14th

IEEE international requirements engineering conference (RE'06). IEEE, pp 59–68
71. Din CY, Rine D (2008) Requirements content goodness and complexity measurement based on NP chunks. VDM Publishing, Saarbrücken
72. Wohlin C, Runeson P, Höst M, Ohlsson MC, Regnell B, Wesslén A (2012) Experimentation in software engineering. Springer, Berlin
73. Landhaußer M, Korner SJ, Tichy WF, Keim J, Krisch J (2015) Denom: a tool to find problematic nominalizations using NLP. In: 2015 IEEE second international workshop on artificial intelligence for requirements engineering (AIRE), pp 1–8. IEEE
74. Mackenzie JL (1985) Nominalization and valency reduction. Predicates and terms in Functional Grammar. Dordrecht/Cinnaminson, Foris, pp 31–51
75. Méndez Fernández D, Penzenstadler B, Kuhrmann M, Broy M (2010) A meta model for artefact-orientation: fundamentals and lessons learned in requirements engineering. In: International conference on model driven engineering languages and systems. Springer, Berlin, pp 183–197
76. Méndez Fernández D, Penzenstadler B (2015) Artefact-based requirements engineering: the Amdire approach. Requir Eng 20(4):405–434
77. Méndez Fernández D, Wieringa R (2013) Improving requirements engineering by artefact orientation. In: International conference on product focused software process improvement. Springer, Berlin, pp 108–122
78. Sharp H, Finkelstein A, Galal G (1999) Stakeholder identification in the requirements engineering process. In: Proceedings. Tenth international workshop on database and expert systems applications. DEXA 99. IEEE, pp 387–391
79. Zhao L, Alhoshan W, Ferrari A, Letsholo KJ, Ajagbe MA, Chioasca E-V, Batista-Navarro RT (2021) Natural language processing for requirements engineering: a systematic mapping study. ACM Comput Surv (CSUR) 54(3):1–41
80. McElreath R (2020) Statistical rethinking: a Bayesian course with examples in R and Stan. Chapman & Hall/CRC, Boca Raton
81. Dybå T, Sjøberg DI, Cruzes DS (2012) What works for whom, where, when, and why? On the role of context in empirical software engineering. In: Proceedings of the ACM-IEEE international symposium on empirical software engineering and measurement, pp 19–28
82. Briand L, Bianculli D, Nejati S, Pastore F, Sabetzadeh M (2017) The case for context-driven software engineering research: generalizability is overrated. IEEE Softw 34(5):72–75
83. Montgomery L, Lüders C, Maalej W (2022) An alternative issue tracking dataset of public jira repositories. In: Proceedings of the 19th International Conference on Mining Software Repositories, pp. 73–77