

# Identifying relevant Factors of Requirements Quality: an industrial Case Study

Julian Frattini<sup>1</sup>[0000–0003–3995–6125]

Blekinge Institute of Technology, Valhallavägen 1, 371 41 Karlskrona, Sweden  
[julian.frattini@bth.se](mailto:julian.frattini@bth.se)

**Abstract.** [Context and Motivation]: The quality of requirements specifications impacts subsequent, dependent software engineering activities. Requirements quality defects like ambiguous statements can result in incomplete or wrong features and even lead to budget overrun or project failure. [Problem]: Attempts at measuring the impact of requirements quality have been held back by the vast amount of interacting factors. Requirements quality research lacks an understanding of which factors are relevant in practice. [Principal Ideas and Results]: We conduct a case study considering data from both interview transcripts and issue reports to identify relevant factors of requirements quality. The results include 17 factors and 11 interaction effects relevant to the case company. [Contribution]: The results contribute empirical evidence that (1) strengthens existing requirements engineering theories and (2) advances industry-relevant requirements quality research.

**Keywords:** Requirements quality · Case study · Interview

## 1 Introduction

Requirements specifications constitute the input to many subsequent software engineering activities and artifacts. Requirements specifications are used to design architecture, develop code, or derive test cases. Hence, the quality of requirements specifications impacts the software engineering process [16,24].

The requirements quality research domain aims to aid practitioners in understanding and managing the quality of their requirements specifications by detecting and removing requirements quality defects [18]. However, empirical evidence in this research domain remains scarce [14]. This hampers the adoption of research results in practice [6,11]. Recent systematic studies of the requirements quality research domain have identified a lack of industrial relevance as a main factor holding the field back [18].

The aim of this study is to contribute empirical evidence about the impact of requirements quality defects by identifying factors of requirements quality that are relevant in practice. To this end, an industrial case study at Ericsson has been conducted. The gathered evidence both strengthens existing requirements engineering theories and steers future research efforts toward solving practically

This preprint has not undergone peer review or any post-submission improvements or corrections. The Version of Record of this contribution is published in “Requirements Engineering: Foundation for Software Quality” (<https://doi.org/10.1007/978-3-031-57327-9>), and is available online at [https://doi.org/10.1007/978-3-031-57327-9\\_2](https://doi.org/10.1007/978-3-031-57327-9_2).

relevant problems. While the results cannot be generalized due to the employed research method, we encourage replication in different industrial contexts with the disclosure of our replication package<sup>1</sup>.

The remainder of this article is structured as follows: Section 2 introduces related work on requirements quality research. Section 3 describes the applied method and Section 4 reports the obtained results. These results are discussed in Section 5 before concluding in Section 6.

## 2 Background

### 2.1 Requirements Quality

Organizations use requirements specifications in several subsequent software engineering activities. Non-functional requirements influence a system’s architecture, functional requirements determine the input and expected output of the system’s features, and all requirements are ultimately translated into test cases to assert whether the developed system meets the customers’ expectations. However, quality defects in requirements specifications like missing or ambiguous requirements impede this reuse [18,24].

Two factors aggravate the impact of requirements quality defects on subsequent activities. Firstly, requirements specifications are predominantly written in natural language [12] (NL). The inherent complexity and ambiguity of NL benefits quality defects. Secondly, the cost to remove a quality defect scales the longer it remains undetected [2]. Clarifying and rewriting an ambiguous requirement takes significantly less time than re-implementing a wrong feature built based on a misunderstood requirement.

Consequently, organizations aim to detect and remove relevant requirements quality defects as early as possible [18]. A popular frame for this is the requirements quality factor. A quality factor is a normative metric that maps a requirements specification to a level of quality [14]. For example, the quality factor *passive voice* associates the use of passive voice in a requirements sentence with bad quality due to the potential omission of the subject of the sentence [21]. Requirements quality research abounds with quality factors and automatic tools to detect violations against them [14].

### 2.2 Requirements Quality Theory

Despite their usability, requirements quality factors suffer from significant shortcomings. Most significantly, the majority of them lack empirical evidence for their relevance [18,14,13], i.e., they are purely normative. Most publications empirically investigate the performance of a tool automatically detecting a violation against a quality factor, but only the fewest empirically investigate whether this violation does have an actual impact and is, therefore, even worth detecting

---

<sup>1</sup> Available at <https://doi.org/10.5281/zenodo.10149475>

and mitigating. This undermines the practical relevance of requirements quality research, and research results are rarely adopted in practice [6,11].

In response, previous research proposed a harmonized *requirements quality theory* (RQT). This theory frames requirements quality as the impact that properties of requirements specifications have on the properties of dependent activities in a given context [8]. Consequently, the RQT does not consider a violation against a quality factor as harmful in itself, but only if this violation has an impact on the activities that use the requirement as an input.

Figure 1 visualizes a reduced version of the RQT [13]. The RQT consists of three groups of concepts: artifact concepts, the context concept, and activity concepts. The concept of an *entity* represents all types of requirements artifacts like use cases or sentences. *Quality factors* are properties of these entities [14], like the length of a specification, the completeness of a use case, or the voice (active or passive) of a single sentence. *Activities* represent every software engineering process that uses a requirements entity as an input [8]. This includes activities like implementing or generating test cases but also more implicit processes like understanding an entity and estimating its effort. *Attributes* are properties of these activities and include metrics like the duration or correctness of an activity. *Context factors* represent properties of the process and involved stakeholders [19], like the domain knowledge of involved engineers, the used process model, or the distribution of the organization. Finally, the *impact* represents the relationship between the quality factors, context factors, and attributes.

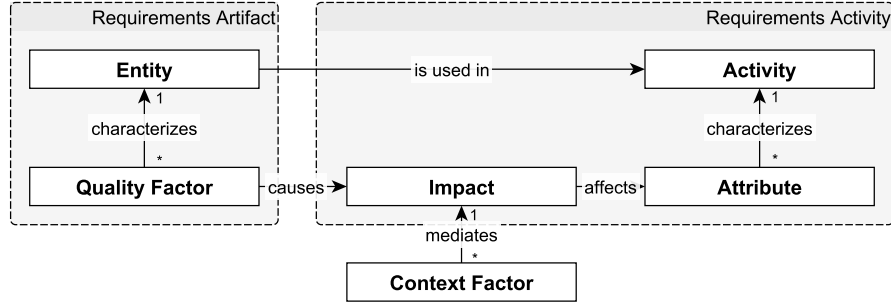


Fig. 1: Groups and concepts of the reduced Requirements Quality Theory [13]

Not only does the RQT guide the framing of requirements quality, but it also enables operationalization in practice [13]. By measuring quality factors, context factors, and attributes, all input and output variables to the impact become quantified. Once all variables are quantified, a statistical model trained on historical data can estimate the probability that a certain quality factor in a given context will affect the attribute of an activity [13]. This quantification was not attempted yet [7] but advocated for in requirements quality research roadmaps [6,13] since it allows to (1) empirically assess and compare the criticality of quality factors, and (2) predict how a requirements specification will

impact dependent activities. This prediction model would meet the initially mentioned need of organizations to reliably detect quality defects that impact the software engineering process.

### 2.3 Gap

Requirements quality research faces two major gaps. Firstly, requirements quality research lacks empirical evidence for the relevance of requirements quality factors [18,14,13]. This entails the risk that requirements quality research does not focus on problems relevant to practice.

Secondly, and by extension, the RQT is difficult to operationalize without empirical evidence about the relevance of quality factors. Previous research has already identified 206 mostly normative requirements quality factors [14]. Measuring all of them is not feasible, given their amount and complexity. Empirical evidence about the relevance of quality factors will aid in prioritizing and selecting factors to consider in statistical models for impact estimation.

## 3 Method

This study aims to address the gaps outlined in Section 2.3 by contributing empirical evidence to relevant factors of requirements quality. The study addresses the following research questions:

- RQ1: Which factors of requirements quality do engineers that process requirements perceive to be relevant?
- RQ2: Which factors of requirements quality are reported in issues?

The study contrasts two perspectives: relevant factors of requirements quality as *perceived* by engineers using requirements (RQ1) and as *reported* in issues (RQ2). The study employs case study research to obtain insights on the necessary level of detail at the expense of generalizability [22]. The contemporary software engineering phenomenon [25] that is subject of the study is the impact of requirements quality defects as described through its factors. The case study research method lends itself to the investigation since the boundary between the phenomenon and the context is unclear [25].

The method follows Runeson et al. [22] and is reported according to the guidelines by Höst et al. [15]. Figure 2 visualizes the steps of the process. Verbatim examples shown in the figure and throughout this section (in quotation marks) are artificial as the raw data is confidential.

### 3.1 Data Collection

Ericsson, the case company providing the data, is a large, globally distributed software development organization. Ericsson follows an agile development approach but completes the *requirements specification* for each new feature or

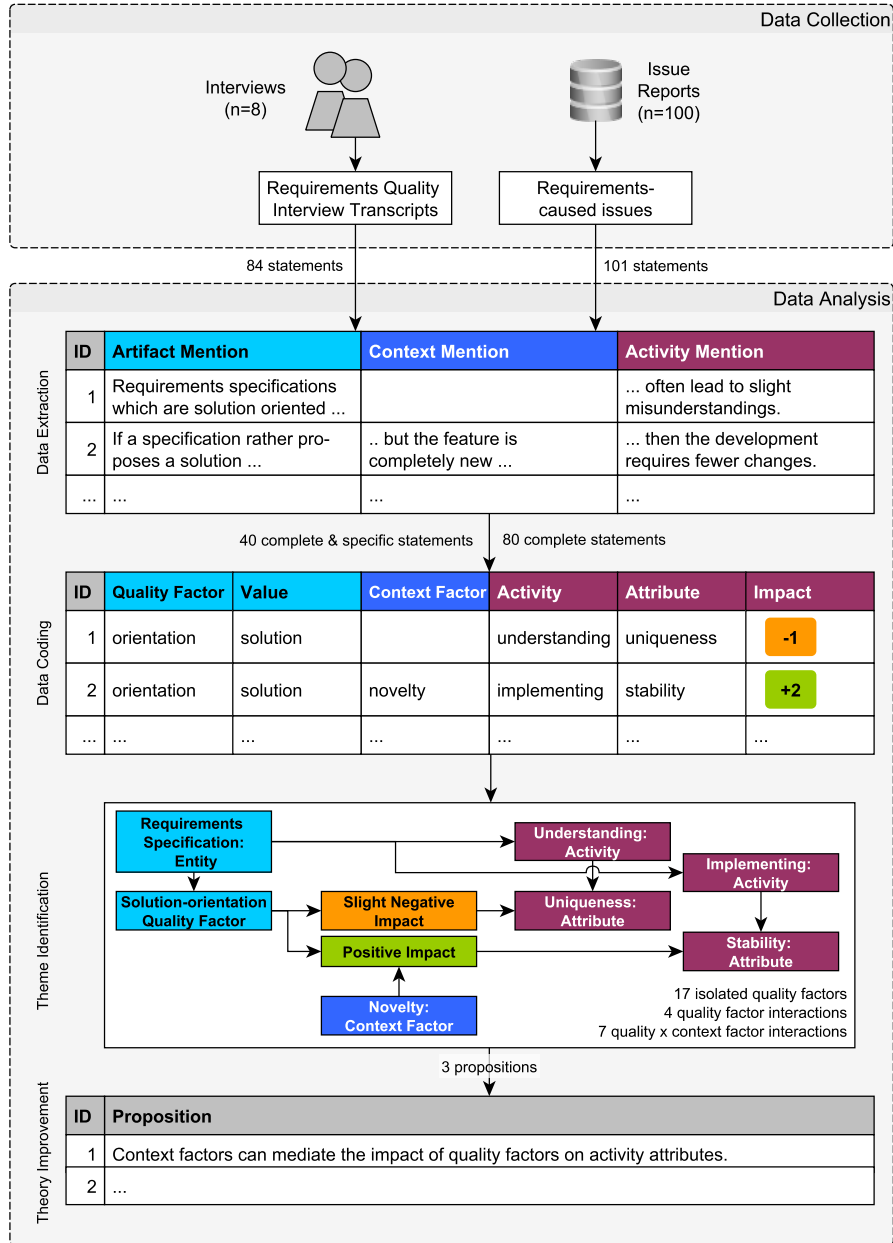


Fig. 2: Overview of the data collection and analysis method

change request before committing to subsequent phases like design, implementation, and testing. These requirements specifications use unconstrained natural language to specify one or more requirements related to a specific feature or change request. The study evaluates two different types of data to triangulate and strengthen the results [25]. The upper section of Figure 2 visualizes the two data collection approaches.

**Interview Data** To understand the factors that engineers processing requirements perceive to be relevant, interviews were conducted (RQ1). A contact at the case company provided a sample of eight software engineers responsible for processing the requirements specification and developing a solution specification. The interview participants had an average of 3.5 years of experience in their role, 7.5 years with the organization, and 15.3 years as software engineers.

A protocol aided in conducting the semi-structured interviews based on previous requirements quality research. In particular, the eleven themes of requirements quality by Montgomery et al. [18] served as sections of the interview. These themes include, for example, ambiguity, completeness, and correctness. For each theme, the interview participants were asked whether they experienced any issues of this type when processing requirements. If yes, they were prompted to elaborate on the issue using the RQT as a frame [13]. Otherwise, the interview moved to the next theme. A slide deck introduced and visualized the RQT to the interview participants to aid the conversation. The author of this paper conducted all eight interviews, each taking up to one hour.

The recorded interview sessions were automatically transcribed using Descript<sup>2</sup>. Afterwards, the author manually checked all transcripts and ensured that the automatic transcription matched the recording. The replication package contains all supplementary material, including the interview guidelines. The transcripts contain confidential information and cannot be shared.

**Issue Data** To understand which factors of requirements quality cause a reported impact, issues from an issue tracker were analyzed (RQ2). The contact at the case company provided access to Ericsson’s database of issue reports. This database contains issues raised both during the internal development process and from external customers. Every issue denotes the development phases in which it was detected and in which the root cause of this issue has been introduced. Domain experts procure and document the latter information. For this study, the available issues were filtered to obtain only those that have been introduced during the requirements engineering phase, resulting in 100 issue reports from January 2021 until September 2023.

---

<sup>2</sup> <https://www.descript.com/>

### 3.2 Data Analysis

To analyze the large body of textual data, thematic synthesis as proposed by Cruzes and Dybå [5] was employed and reported according to their guideline. The lower section of Figure 2 visualizes the data analysis steps.

**Data extraction** Extracting relevant data from the textual corpus comprised the first step. Each defect perceived by an interview participant constitutes one statement about requirements quality. The eight interview transcripts produced 84 statements about perceived requirements quality. The 100 issues contained 101 statements about reported requirements quality. The three groups of the RQT (artifact, context, and activity) provided a frame for the data extraction. For each of the three groups, all relevant mentions from a statement were extracted. For example, the statement “If a specification rather proposes a solution, but the feature is completely new, then the development requires fewer changes.” contains one mention of each group: “a specification rather proposes a solution” describes a property of the artifact, “the feature is completely new” describes the context, and “the development requires fewer changes” represents the activity impacted by the quality and context factor.

**Data coding** Each mention received codes for all the concepts contained within the respective RQT group. An artifact mention can contain a number of quality factors with a value associated with each of them. A context mention can contain a number of context factors. An activity mention can contain a number of activities, each associated with one attribute and an impact value. All codes and coding instructions were documented in extensive coding guidelines.

Coding the artifact and context mentions followed a deductive approach [17]: quality factors [14,18] and context factors [19] identified during previous research constituted the available codes. Coding the activity mentions followed an inductive approach [23] since literature is still lacking in this regard [6,13]. Figure 3 visualizes the activity and attribute codes generated inductively from the data. Descriptions of all activities and attributes can be found in the coding guideline contained in the replication package.

The artifact mention “if a specification rather proposes a solution” received the quality factor code *orientation* and the value *solution*, since it describes a *solution-oriented* requirements specification. The context factor mention “but the feature is completely new” was coded *novelty*. Finally, the activity mention “then the development requires less changes” received the following codes: the *stability* attribute of the *implementing* activity experiences a *positive* impact (+2). We coded the strength of the impact on a discrete seven-point scale. Positive (+2), negative (-2), and no impact (±0) were used as the default codes depending on the direction of the impact. Particularly strong (+3 and -3) or weak (+1 and -1) codes were only used when explicitly mentioned by the interview participant, e.g., if an impact was “very critical” or just “slight.”

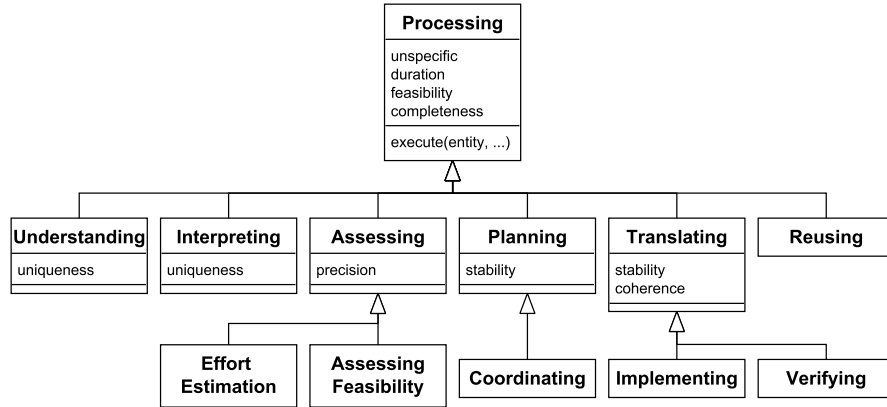


Fig. 3: Model of activities (as classes) and attributes (as their attributes)

Independent coders were involved to ensure the reliability of the subjective coding task. For the coding of the interview transcripts, a senior researcher from the same department as the author, who is also familiar with the case company, independently coded ten randomly selected statements using the coding guideline. The inter-rater agreement of codes achieved a percentage score of 83.8% and a Cohen’s Kappa of 71.8%. However, Cohen’s Kappa is known to be unreliable for uneven marginal distributions [9]. Still, the more robust S-Score [1] resulted in an agreement of 82.3%. All scores represent a substantial agreement and support the reliability of the coding process.

For the coding of the issue reports, a senior engineer from the case company was involved. In a session between the author and the involved engineer, questionable codes were reviewed and adjusted. This process confirmed the applicability of the chosen codes as well as that 21 of the selected issues were not caused by requirements quality defects but unrelated circumstances. These 21 statements were not considered in future phases of the data analysis.

**Theme identification** In this step, individual codes are aggregated into “more meaningful and parsimonious units” [5]. The theme identification imposes two further conditions on the coded statements: (C1) A statement must contain at least one quality factor and at least one activity, and (C2) a statement must not contain any unspecific activity (activity code: *processing*, root of Figure 3) or an unspecific attribute (attribute code: *unspecific*). This rules out 44 vague interview statements of the form “*quality factor* is bad” (C1, missing activity) or “*quality factor* is bad for implementing” (C2, unspecific attribute).

The final analysis considered 40 interview and 80 issue statements. The amount of information per mention differed between the two data sources. While the interview encouraged participants to elaborate on all concepts of the RQT, the issue data lacked the same level of control [22]. Issue statements always con-



tained the activity concepts since they explicitly report the effect of an issue, but the level of information on the root cause in requirements engineering was often limited, and the issues did not contain any information about the context.

The analysis of the more granular and complete interview statements splits the data into three groups: statements containing a single quality factor and no context factors, statements containing multiple quality factors but no context factors, and statements containing context factors. Within each group, statements about the same quality factors are aggregated to collect all impacted activities and attributes. Since the issue statements contained codes on higher levels and no context factors, they were aggregated into one matrix showing the distribution of quality factors impacting activity attributes.

**Model creation** The final inferential step of the guideline by Cruzes et al. is the description of higher-order themes, a taxonomy, model, or a theory [5]. Because this study is already grounded in the theoretical foundation provided by the RQT [13], but none of the encountered data challenged this theory, developing a new model or theory was not deemed constructive. Instead, this study evolves the existing RQT by deriving propositions from the identified themes. These propositions enrich the theory with empirical insights and contribute falsifiable hypotheses for further research.

**Trustworthiness Assessment** The final overall step of the guideline by Cruzes et al. is to assess the trustworthiness of the synthesis [5]. As these concerns align with threats to validity, they are addressed in Section 5.2.

## 4 Results

### 4.1 Interview data

**Impact of single Quality Factors** The interview data contains information about 17 unique quality factors with an impact on at least one subsequent activity. For brevity, Table 1 lists only the four quality factors that were contained in at least two statements. Each cell of the *impact* column in Table 1 lists the perceived direction and strength of the impact that the quality factor has on the attribute of the activity in this cell. For example, two statements of the interview data described a negative impact of solution-oriented requirements on a unique understanding of that requirement. One statement stressed that this impact is major (-3), the other one did not (-2). The replication package contains the remaining quality factors and their impact<sup>3</sup>.

The table shows that the interview participants perceived the four most often mentioned quality factors to impact a variety of activities and their attributes.

<sup>3</sup> Available at <https://github.com/JulianFrattini/rqi-relf/blob/main/src/analytics/results.md>

Table 1: Perceived impact of single quality factors on subsequent activities

Quality Factor	Activity	Attribute	Impact
Solution-orientation	Understanding	Uniqueness	-3 -2
	Verifying	Completeness	-2 -2
	Effort Estimation	Traceability	+2
	Translating	Stability	+2
	Feasibility Assessment	Precision	+2
	Planning	Stability	+2
Non-atomic	Translating	Duration	-2 -2
	Planning	Stability	-2
Non-concise	Understanding	Uniqueness	-1
		Duration	-2
Too dense	Understanding	Duration	-2
	Interpreting	Uniqueness	-2
	Verifying	Duration	-2

The most frequently mentioned quality factor is *solution-orientation*, i.e., requirements that impose on the solution space rather than elaborating on the problem space [10]. This quality factor is perceived to cause misunderstandings (i.e., causing the *understanding* activity to be not *unique*) and lack of coverage when deriving test cases (i.e., causing the *verifying* activity to be not *complete*). The table also shows that some quality factors have a mixed impact on different activities. For example, a solution-oriented requirements specification is also perceived to aid effort estimation, translating, and planning.

**Interaction of multiple Quality Factors** The interview data contains information about four unique interactions between quality factors. Table 2 lists the four interaction effects. The interview participants reported that redundant requirements that were also not connected through horizontal trace links (i.e., links between requirements) caused incoherent implementations. Furthermore, non-functional requirements were reported to be susceptible to ambiguous understanding when providing to little details. Additionally, requirements specifications that were yet immature but also already committed to were quicker to implement due to the applied time pressure (+2), but implementation became much less feasible (-3). Finally, the precision of feasibility assessment suffered from jargonistic and dense requirements.

**Interaction with Context Factors** The interview data contains information about seven unique interactions between quality factors and context factors. The two most prominent prominently perceived interaction effects involve the quality factor *solution-orientation* and *density*. Figure 4 visualizes one statement describing the interaction between the quality factor *solution-orientation* and

Table 2: Interaction between two quality factors

Quality Factor 1	Quality Factor 2	Activity	Attribute	Impact
Redundancy	Missing horizontal trace links	Implementing	Coherence	-2
Too little details	Non-functional requirement	Understanding	Uniqueness	-2
Immature	Committed	Implementing	Duration	+2
			Feasibility	-3
Jargonic	Too dense	Assessing Feasibility	Precision	-2

context factor *involvement* on the *uniqueness* attribute of the *understanding* activity. If the stakeholder responsible for processing the requirement was also involved in writing it, the impact on the understandability is mitigated. The remaining interaction effects are detailed in our replication package but follow a similar pattern: Context factors like *involvement*, *experience*, and *supplementary communication* can mitigate the negative impact of quality factors. Additionally, solution-oriented requirements exhibit an even stronger positive impact on several activities like feasibility assessment and effort estimation when the context of the requirement is *new*.

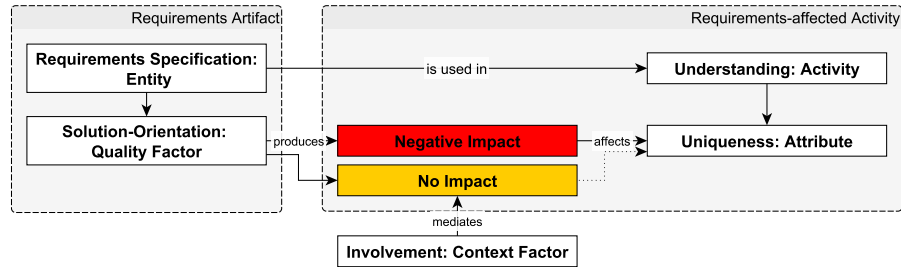


Fig. 4: Interaction effect between *solution-orientation* and *involvement* on the *understanding* activity

## 4.2 Issue reports

The issue data only allows to infer general relationships between quality factors and activities. Table 3 lists the number of statements per constellation of quality factor, activity, and attribute. The most prominent impact of requirements quality that results in a reported issue is *completeness*, i.e., caused by a missing requirement. This results mostly in *incorrect* (i.e., bugs) or *incomplete* (i.e., missing features) implementations. In rare cases, the *understanding*, *interpreting*, and *verifying* activity are reported to be impacted. Behind completeness, the most often reported impact is *consistency* and *ambiguity*.

Table 3: Requirements impact as recorded in the issue reports (Comp. = Completeness, Corr. = Correctness, Cons. = Consistency)

<b>Activity Attribute</b>	Understanding	Interpreting	Implementing			Verifying	
	Unique	Unique	Comp.	Corr.	Cons.	Coverage	Feasible
<b>Quality Factor</b>							
Completeness	1	0	13	34	3	1	1
Consistency	1	0	0	6	2	2	0
Ambiguity	1	2	2	1	2	0	0
Correctness	1	0	0	2	1	0	0
Feasibility	0	0	0	1	0	1	0
Relevance	0	0	1	0	0	0	0

### 4.3 Propositions

The triangulation of interview and issue analysis results allows to derive the following propositions enhancing the RQT.

**Relevant quality factors** The set of quality factors perceived and reported to be relevant in the case company is very limited. Among the perceived quality factors, only solution-orientation, lack of atomicity, lack of conciseness, and density received support in at least two statements. Among the reported quality factors, lack of completeness (i.e., missing requirements) stands out as the primary cause of issues in the down-stream development process.

**Mixed Impact** The analysis of the interview data shows that quality factors can have mixed impact on different activities. For example, while a solution-oriented requirement might negatively impact the activities of understanding and verifying it, planning and translating activities become more stable.

**Interactions matter** The analysis of the interview data shows that interactions between quality factors but also between quality and context factors have a significant effect. In particular, quality factors like the novelty of a feature, the experience of the involved engineers, and supplementary communication mediate the effect of quality factors as shown in Figure 4.

## 5 Discussion

### 5.1 Implications

**Contribution to Research** The empirical evidence both strengthens existing requirements engineering theories and guides further advances in requirements

quality research. The results strengthen the RQT [13,8] in that artifact properties impacting activity properties constitutes requirements quality. Moreover, the results confirm that one quality factor may have different impacts on different activities [8]. Incautiously removing a quality factor from an entity due to the negative impact on one activity may, therefore, also mitigate its positive impact on other activities. The results further strengthen the *Naming the Pain in Requirements Engineering* (NaPiRE) initiative [16] by contributing more granular evidence to the problems of RE relevant to practice. The results agree with the conclusion of NaPiRE that missing requirements are among the most impactful quality defects. Similarly, the results of both data sources of the study agree with previous studies that the effect of ambiguity is less relevant in practice than often assumed [20,4]. Finally, the results support the advocacy for context-sensitive research in empirical software engineering [3] by emphasizing context factors as mediators of the impact of artifacts on activities. The results of this study guide further research advances through the approach of identifying relevant factors of requirements quality. The study reduces the vast space of several hundred potential quality factors [14] down to about 30 that are relevant to the specific context of an organization.

**Impact on Practice** The operationalization of the RQT in practice becomes feasible due to the reduction of variables to measure. The results steer the next step of research with the case company toward detecting solution-oriented, non-atomic, non-concise, dense, and incomplete requirements. Additionally, effort will be focused on measuring the relevant context factors of involvement, experience, novelty, and supplementary communication. These measurements enable the impact estimation of requirements quality on dependent activities and advance requirements quality research roadmaps [6,13].

**Limitations** One gap to overcome is the lack of an overview of requirements-dependent activities and their measurable attributes, as also outlined in previous research roadmaps [6,13]. These attributes constitute the dependent variable in the impact estimation implied by the RQT [13]. The interview data contains 44 statements with either an unclear activity or attribute impacted by a quality factor. Developing a model of requirements-dependent activities and their attributes is a necessary next step to achieve operationalization of the RQT [13].

## 5.2 Threats to Validity

This section discusses threats to validity according to Runesson et al. [22] and Wohlin et al. [26] and, additionally, addresses concerns of trustworthiness of the thematic synthesis [5].

Regarding conclusion validity, the *reliability of measures* is a prevalent threat given the subjective coding process but has been minimized by involving independent raters and calculating inter-rater agreement where applicable. Similarly,

the lack of control over the issue data questions their reliability. Involving a senior engineer from the case company mitigated this issue, providing an adequate *confirmability* of the data.

Regarding internal validity, the interview data suffers from *selection* bias. The interview participants were sampled by the industry contact. However, the participants show a wide variation of background and experience, which leads to assume that this threat is minimal.

Regarding construct validity, the study suffers from *inadequate preoperational explication of constructs*, i.e., immaturity of some of the concepts of the theoretical framework (the RQT). In particular, the activity group within the RQT (activities and attributes) is insufficiently explored in requirements quality research [6,13]. As a consequence, several interview statements failed to specify the activity and attribute impacted by requirements quality. The threat was minimized by excluding this data from the analysis.

Regarding external validity, the inference of this study is not generalizable or transferable [5] by design of the case study method. Additional research replicating this study in other companies are necessary to generalize the results.

## 6 Conclusion

This case study demonstrates the application of the requirements quality theory (RQT) to identify relevant factors of requirements quality. By analyzing both interview and issue report data, we identified 17 relevant quality as well as 11 interaction effects among them and with context factors. The study contributes empirical evidence to the relevance of these factors and their effects in the case company. The study emphasizes that (1) some requirements quality factors are more relevant in practice than others, (2) they may have a simultaneous negative and positive impact on different activities, and (3) context factors mediate their impact. This research advances requirements quality research by advancing existing research roadmaps [6] toward a quantified impact estimation of requirements quality in practice [13].

## References

1. Bennett, E.M., Alpert, R., Goldstein, A.: Communications through limited-response questioning. *Public Opinion Quarterly* **18**(3), 303–308 (1954)
2. Boehm, B.W., Papaccio, P.N.: Understanding and controlling software costs. *IEEE transactions on software engineering* **14**(10), 1462–1477 (1988)
3. Briand, L., Bianculli, D., Nejati, S., Pastore, F., Sabetzadeh, M.: The case for context-driven software engineering research: generalizability is overrated. *IEEE Software* **34**(5), 72–75 (2017)
4. de Bruijn, F., Dekkers, H.L.: Ambiguity in natural language software requirements: A case study. In: *Requirements Engineering: Foundation for Software Quality: 16th International Working Conference, REFSQ 2010, Essen, Germany, June 30–July 2, 2010. Proceedings* 16. pp. 233–247. Springer (2010)

5. Cruzes, D.S., Dyba, T.: Recommended steps for thematic synthesis in software engineering. In: 2011 international symposium on empirical software engineering and measurement. pp. 275–284. IEEE (2011)
6. Femmer, H.: Requirements quality defect detection with the qualicen requirements scout. In: REFSQ Workshops (2018)
7. Femmer, H., Fernández, D.M., Wagner, S., Eder, S.: Rapid quality assurance with requirements smells. *Journal of Systems and Software* **123**, 190–213 (2017)
8. Femmer, H., Mund, J., Fernández, D.M.: It’s the activities, stupid! a new perspective on re quality. In: 2015 IEEE/ACM 2nd International Workshop on Requirements Engineering and Testing. pp. 13–19. IEEE (2015)
9. Feng, G.C.: Mistakes and how to avoid mistakes in using intercoder reliability indices. *Methodology: European Journal of Research Methods for the Behavioral and Social Sciences* **11**(1), 13 (2015)
10. Fernandez, D.M., Wagner, S., Lochmann, K., Baumann, A., de Carne, H.: Field study on requirements engineering: Investigation of artefacts, project parameters, and execution strategies. *Information and Software Technology* **54**(2), 162–178 (2012)
11. Franch, X., Mendez, D., Vogelsang, A., Heldal, R., Knauss, E., Oriol, M., Travassos, G., Carver, J.C., Zimmermann, T.: How do practitioners perceive the relevance of requirements engineering research? *IEEE Transactions on Software Engineering* (2020)
12. Franch, X., Palomares, C., Quer, C., Chatzipetrou, P., Gorschek, T.: The state-of-practice in requirements specification: an extended interview study at 12 companies. *Requirements Engineering* pp. 1–33 (2023)
13. Frattini, J., Montgomery, L., Fischbach, J., Mendez, D., Fucci, D., Unterkalmsteiner, M.: Requirements quality research: a harmonized theory, evaluation, and roadmap. *Requirements engineering* (2023)
14. Frattini, J., Montgomery, L., Fischbach, J., Unterkalmsteiner, M., Mendez, D., Fucci, D.: A live extensible ontology of quality factors for textual requirements. In: 2022 IEEE 30th International Requirements Engineering Conference (RE). pp. 274–280. IEEE (2022)
15. Host, M., Runeson, P.: Checklists for software engineering case study research. In: First international symposium on empirical software engineering and measurement (ESEM 2007). pp. 479–481. IEEE (2007)
16. Méndez, D., Wagner, S., Kalinowski, M., Felderer, M., Mafra, P., Vetrò, A., Conte, T., Christiansson, M.T., Greer, D., Lassenius, C., et al.: Naming the pain in requirements engineering: Contemporary problems, causes, and effects in practice. *Empirical software engineering* **22**, 2298–2338 (2017)
17. Miles, M.B., Huberman, A.M.: *Qualitative data analysis: An expanded sourcebook*. sage (1994)
18. Montgomery, L., Fucci, D., Bouraffa, A., Scholz, L., Maalej, W.: Empirical research on requirements quality: a systematic mapping study. *Requirements Engineering* **27**(2), 183–209 (2022)
19. Petersen, K., Wohlin, C.: Context in industrial software engineering research. In: 2009 3rd International Symposium on Empirical Software Engineering and Measurement. pp. 401–404. IEEE (2009)
20. Philippo, E.J., Heijstek, W., Kruiswijk, B., Chaudron, M.R., Berry, D.M.: Requirement ambiguity not as important as expected—results of an empirical evaluation. In: *Requirements Engineering: Foundation for Software Quality: 19th International Working Conference, REFSQ 2013, Essen, Germany, April 8–11, 2013. Proceedings* 19. pp. 65–79. Springer (2013)

21. Rosadini, B., Ferrari, A., Gori, G., Fantechi, A., Gnesi, S., Trotta, I., Bacherini, S.: Using nlp to detect requirements defects: An industrial experience in the railway domain. In: Requirements Engineering: Foundation for Software Quality: 23rd International Working Conference, REFSQ 2017, Essen, Germany, February 27–March 2, 2017, Proceedings 23. pp. 344–360. Springer (2017)
22. Runeson, P., Host, M., Rainer, A., Regnell, B.: Case study research in software engineering: Guidelines and examples. John Wiley & Sons (2012)
23. Strauss, A., Corbin, J.: Basics of qualitative research. Sage publications (1990)
24. Wagner, S., Fernández, D.M., Felderer, M., Vetrò, A., Kalinowski, M., Wieringa, R., Pfahl, D., Conte, T., Christiansson, M.T., Greer, D., et al.: Status quo in requirements engineering: A theory and a global family of surveys. *ACM Transactions on Software Engineering and Methodology (TOSEM)* **28**(2), 1–48 (2019)
25. Wohlin, C.: Case study research in software engineering—it is a case, and it is a study, but is it a case study? *Information and Software Technology* **133**, 106514 (2021)
26. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in software engineering. Springer Science & Business Media (2012)