

Rapid Quality Assurance with Requirements Smells

Henning Femmer^{a,*}, Daniel Méndez Fernández^a, Stefan Wagner^b, Sebastian Eder^a

^a*Software & Systems Engineering, Technische Universität München, Germany*

^b*Institute of Software Technology, University of Stuttgart, Germany*

Abstract

Context: Bad requirements quality can cause expensive consequences during the software development lifecycle, especially if iterations are long and feedback comes late. **Objectives:** We aim at a light-weight static requirements analysis approach that allows for rapid checks immediately when requirements are written down. **Method:** We transfer the concept of *code smells* to Requirements Engineering as *Requirements Smells*. To evaluate the benefits and limitations, we define Requirements Smells, realize our concepts for a smell detection in a prototype called *Smella* and apply *Smella* in a series of cases provided by three industrial and a university context. **Results:** The automatic detection yields an average precision of 59% at an average recall of 82% with high variation. The evaluation in practical environments indicates benefits such as an increase of the awareness of quality defects. Yet, some smells were not clearly distinguishable. **Conclusion:** Lightweight smell detection can uncover many practically relevant requirements defects in a reasonably precise way. Although some smells need to be defined more clearly, smell detection provides a helpful means to support quality assurance in Requirements Engineering, for instance, as a supplement to reviews.

Keywords: Requirements Engineering, Quality Assurance, Automatic Defect Detection, Requirements Smells

Contents

1 Introduction

2 Related work

- 2.1 The notion of smells in software engineering
- 2.2 Quality assurance of software requirements
- 2.3 Discussion

3 Requirements Smells

- 3.1 Requirements Smell terminology
- 3.2 Requirements Smells based on ISO 29148

4 Smella: A prototype for Requirements Smell detection

4.1 Requirements parsing	11
4.2 Language annotation	11
2 4.3 Findings identification	12
3 4.4 Findings presentation	12
5 Requirements Smell detection in the process of quality assurance	15
3 6 Evaluation	16
5 6.1 Case study design	16
6.1.1 Research questions	16
6.1.2 Case and subjects selection	17
6.1.3 Data collection procedure	17
6.1.4 Analysis procedure	18
6.1.5 Validity procedure	19
6.2 Results	19
6.2.1 Case and subjects description	20

6.2.2	RQ 1: How many Requirements Smells are present in the artifacts?	21
6.2.3	RQ 2.1: How accurate is the smell detection?	25
6.2.4	RQ 2.2: Which of these smells are practically relevant in which context?	29
6.2.5	RQ 3: Which requirements quality defects can be detected with smells?	30
6.2.6	RQ 4: How could smells help in the QA process?	33
6.2.7	Evaluation of validity	34

7 Conclusion

7.1	Summary of conclusions
7.2	Relation to existing evidence
7.3	Impact/Implications
7.4	Limitations
7.5	Future work

Appendix A Requirements Checklist 41

1. Introduction

Defects in requirements, such as ambiguities or incomplete requirements, can lead to time and cost overruns in a project [56]. Some of the issues require specific domain knowledge to be uncovered. For example, it is very difficult to decide whether a requirements artifact is complete without domain knowledge. Other issues, however, can be detected more easily: If a requirement states that a sensor should work with *sufficient accuracy* without detailing what *sufficient* means in that context, the requirement is vague and consequently not testable. The same holds for other pitfalls such as loopholes: Phrasing that a certain property of the software under development should be fulfilled *as far as possible* leaves room for subjective (mis-)interpretation and, thus, can have severe consequences during the acceptance phase of a product [24, 33].

To detect such quality defects, quality assurance processes often rely on reviews. Reviews of requirements artifacts, however, need to involve all relevant

stakeholders [65], who must manually read and understand each requirements artifact. Moreover, they are difficult to perform. They require a high domain knowledge and expertise from the reviewers [65] and the quality of their outcome depends on the quality of the reviewer [75]. On top of all this, reviewers could be distracted by superficial quality defects such as the aforementioned vague formulations or loopholes. We therefore argue that reviews are time-consuming and costly.

Therefore, quality assurance processes would benefit from faster feedback cycles in requirements engineering (RE), which support requirements engineers and project participants in immediately discovering certain types of pitfalls in requirements artifacts. Such feedback cycles could enable a lightweight quality assurance, e.g., as a complement to reviews.

Since requirements in industry are nearly exclusively written in natural language [58] and natural language has no formal semantics, quality defects in requirements artifacts are hard to detect automatically. To face this challenge of fast feedback and the imperfect knowledge of a requirement's semantics, we created an approach that is based on what we call *Requirements (Bad) Smells*. These are concrete symptoms for a requirement artifact's quality defect for which we enable rapid feedback through automatic smell detection.

In this paper, we contribute an analysis of whether and to what extent Requirements Smell analysis can support quality assurance in RE. To this end, we

1. define the notion of *Requirements Smells* and integrate the Requirements Smells¹ concept into an analysis approach to complement (constructive and analytical) quality assurance in RE,
2. present a prototypical realization of our smell detection approach, which we call *Smella*, and
3. conduct an empirical investigation of our approach to better understand the usefulness of

¹In context of our studies, we use the ISO/IEC/IEEE 29148:2011 standard [33] (in the following: *ISO 29148*) as basis for defining requirements quality. The standard supplies a list of so-called *Requirements Language Criteria*, such as loopholes or ambiguous adverbs, which we use to define eight smells (see also the smell definition in Sect. 3.2).

a Requirements Smell analysis in quality assurance.

Our empirical evaluation involves three industrial contexts: The companies Daimler AG as a representative for the automotive sector, Wacker Chemie AG as a representative for the chemical sector, and TechDivison GmbH as an agile-specialized company. We complement the industrial contexts with an academic one, where we apply Smella to 51 requirements artifacts created by students. With our evaluations, we aim at discovering the accuracy of our smell analysis taking both a technical and a practical perspective that determines the context-specific relevance of the detected smells. We further analyze which requirements quality defects can be detected with smells, and we conclude with a discussion of how smell detection could help in the (industrial) quality assurance (QA) process.

Previously published material

This article extends our previously published workshop paper [24] in the following aspects: We provide a richer discussion on the notion of Requirements Smell and give a precise definition. We introduce our (extended) tool-supported realization of our smell analysis approach and outline its integration into the QA process. We extend our first two case studies with another industrial one as well as with an investigation in an academic context to expand our initial empirical investigations by

1. investigating the accuracy of our smell detection including precision, recall, and relevance from a practical perspective,
2. analyzing which quality defects can be detected with smells and
3. gathering practitioner's feedback on how they would integrate smell detection in their QA process considering both formal and agile process environments.

Outline

The remainder of this paper is structured as follows. In Sect. 2, we describe previous work in the area. In Sect. 3, we define the concept of Requirements Smells and describe how we derived a set of Requirements

Smells from ISO 29148. We introduce the tool realization in Sect. 4 and discuss the integration of smell detection in context of quality assurance in Sect. 5. In Sect. 6, we report on the empirical study that we set up to evaluate our approach, before concluding our paper in Sect. 7.

2. Related work

In the following, we discuss work relating to the concept of natural language processing and smells in general, followed by quality assurance in RE, before critically discussing currently open research gaps.

2.1. The notion of smells in software engineering

The concept of code smells was, to the best of our knowledge, first proposed by Fowler and Beck [27] to answer the question at which point the quality of code is so low that it must be refactored. According to Fowler and Beck, the answer cannot be objectively measured, but we can look for certain concrete, visible symptoms, such as duplicated code [27] as an indicator for bad maintainability [35]. This concept of smells, as well as the list that Fowler and Beck proposed, led to a large field of research. Zhang et al. [76] provide an in-depth analysis of the state of the art in code smells. The metaphor of smells as concrete symptoms has since then been transferred to quality of other artifacts including (unit) test smells [72] and smells for system tests in natural language [31]. Cierniewska et al. [12], further characterize different defects of use cases through the term use case smell. In our work, we extend the notion of smells to the broader context of requirements engineering and introduce a concrete definition for the term *Requirements Smell*.

2.2. Quality assurance of software requirements

The concept of Requirements Smells is located in the context of RE quality assurance (QA), which is performed either manually or automatically.

Manual QA. Various authors have worked on QA for software requirements by applying manual techniques. Some put their focus on the classification of quality into characteristics [15], others develop comprehensive checklists [2, 6, 50, 39, 38]. Regarding QA, some

develop constructive QA approaches, such as creating new RE languages, e.g. [17], to prevent issues up front, others develop approaches to make analytic QA, such as reviews, more effective [69]. In a recent empirical study on analytical QA, Parachuri et al. [60] manually investigate the presence of defects in use cases. To sum it up, these works on manual QA provide analytical and constructive methods, as well as (varying) lists for defects. They strengthen our confidence that today’s requirements artifacts are vulnerable to quality defects.

Automatic QA. Various publications discuss the automatic detection of quality violations in RE. We summarize existing approaches and tools, their publications, and empirical evaluations in Table 2. We also created an in-depth analysis of in total 27 related publications evaluating which quality defects or smells the approaches opt for in their described detection. In the following, we will first explain two related areas (automatic QA for redundancy and for controlled languages), before discussing automatic QA for ambiguity in general. For ambiguity, we first describe those approaches that conducted empirical evaluations of precision or recall of quality defects related, but not identical to, the ones of ISO 29148. Afterwards, we focus on publications that mention the same criteria as in the ISO 29148 (see Table 1 for this list and their respective empirical evaluations) and discuss the chosen approaches and results. We publish the complete list of each quality defect that is detected by each of the 27 papers, as well as the precision and recall (where provided), online as supplementary material [25].

Automatic QA for redundancy. One specific area of QA is avoiding redundancy and cloning. Whereas Juergens et al. [34] use ConQAT to search for syntactic identity resulting from a copy-and-paste reuse, Falessi et al. [21] aim at detecting similar content, therefore using methods from information retrieval (such as Latent Semantic Analysis [52]). Rago et al. [62] extend this work specifically for use cases. Their tool, ReqAlign, classifies each step with a semantic abstraction of the step. These publications analyze the performance of their approaches, and de-

pending on the artifact and methods achieve precision and recall close to 1 (see Table 2).

Automatic QA for controlled languages. Another specific area is the application of controlled language and the QA of controlled language. RETA [4] specifically analyzes requirements that are written via certain requirements patterns (such as with the EARS template [54]). Their goal is to detect both conformance to the template but also some of the ambiguities as defined by Berry et al [7]. The authors report on a case study where they look at the template conformance in depth, indicating that template conformance can be classified with various NLP suites to a high accuracy (Precision > 0.85, Recall > 0.9), both with and without glossaries. However, the performance of ambiguity detection (such as the detection of pronouns) is not further discussed in the publication. Similarly, AQUSA [51] analyzes requirements written in user story format (c.f. [14] for a detailed introduction into user stories), and detects various defects, such as missing rationales, where they achieve a precision of 0.63–1. Circe [1, 29] is a further tool that assumes that requirements are written in such requirements patterns and detects violations of context- and domain-specific quality characteristics by building logical models. The authors report on six exemplary findings, which were detected in a NASA case study. However, despite their value to automatic QA, such approaches require very specific requirements structure.

Automatic QA for ambiguity in general. The remaining approaches listed in Table 2 aim at detecting ambiguities in unconstraint natural language. Since the quality defects detected by the approaches by Cierniewska et al. [12], Kof [45], HeRA by Knauss et al. [41, 42], Kiyavitskaya et al. [40], RESI by Körner et al. [46, 47, 48], and Alpino by DeBruijn et al. [16] are not the ones discussed in ISO 29148 and since we could not find an evaluation of precision and recall of these approaches, we omit discussing these approaches in-depth here. An analysis of what these approaches focus on in detail as well as their evaluation can be found in short in Table 2 and in full length in our supplementary material online [25]. In the following, we first report on those publications that focus on criteria

different from ISO 29148, but which report precision or recall. Afterwards, we describe publications that aim at detecting quality violations of ISO 29148 (see Table 1).

First, Chantree et al. [11] target the specific grammatical issue of coordination ambiguity (detecting problems of ambiguous references between parts of a sentence), mostly through statistical methods, such as occurrence and co-occurrence of words. In a case study, they report on a precision of their approach mostly between 54% and 75%. even though they do not explicitly differentiate between the detected ambiguities and the concept of pronouns. Second, Gleich et al. [30] base their approach on the ambiguity handbook, as defined by Berry et al. [7], as well as company-specific guidelines. They compare their dictionary- and POS-based approach against a gold standard which they created by letting people highlight ambiguities in requirements sentences. The gold standard deviates substantially, however, from what is considered high quality in their guidelines. Therefore, they create an additional gold standard, mostly based on the guideline rules. Consequently, their precision² varies between 34% for the pure experts opinion, to 97% for a more guideline-based gold standard. Third, Krisch and Houdek [49], focus on the detection of passive voice and so-called weak words. They present their dictionary- and POS-based approach to practitioners and find many false positives, similar to our RQ 3. In average, a precision of 12% is reported for the weak words detection. These approaches focus on very related, but not identical quality violations or smells.

Automatic QA for ISO 29148 criteria. Lastly, we specifically focus on those approaches that report to detect the criteria from the ISO 29148 standard. Table 1 provides an overview of these works and their respective evaluations.

²Gleich et al. calculate their metrics based on the combination of all ambiguities; unfortunately, they do not differentiate, e.g. by the type of ambiguity. Also, to our knowledge, the gold standard does not differentiate between the types. This prevents a direct comparison to their work.

The ARM tool [74] defines quality in terms of the (now superseeded) IEEE 830 standard [32] and proposes generic metrics, instead of giving feedback directly to requirements engineers. The metrics are calculated through counting how often a set of pre-defined terms (per metric) occurs in a document, including a metric of what we call Loopholes. Even though they report on a case study with 46 specifications from NASA, only a quantitative overview is reported³. The QuARS tool [19, 18] is based on the author’s experience. Bucciarone et al. [10] describe the use of QuARS in a case study with Siemens and show some exemplary findings. SyTwo [22] adopts the quality model of QuARS and applies it to use cases. Loopholes and Subjectivity are part of the QuARS quality model. Also RQA is built on a different, proprietary quality model, as described by Génova et al. [28], which includes negative terms as well as pronouns as quality defects. These works also built upon extending natural language with NLP annotations, such as POS tags and searching through dictionaries for certain problematic phrases. However, we could not find a detailed empirical investigation of these tools, e.g. with regards to precision and recall. SREE is an approach by Tjong and Berry [70], which aims at detection of ambiguities with a recall of 100%. Therefore, they completely avoid all NLP approaches (since they come with imprecision), and build large dictionaries of words. The tool includes detection of loopholes, as well as pronouns; however, they report only on an aggregated precision for all the different types of ambiguities (66-68%) from two case studies. In our previous paper [24], we searched for violations of ISO 29148, yet we provided only a quantitative analysis, as well as qualitative examples. As mentioned before, RETA also issues warnings for pronouns, however, the evaluation in their paper [4] focusses on template conformance.

2.3. Discussion

Previous work has led to many valuable contributions to our field. To explore open research gaps, we now critically reflect on previous contributions from

³See also our RQ 1 in Sect. 6.

Table 1: Related work on criteria of ISO-29148 standard, detailed supplementary material can be found online [25]

	ARM [74]	[19]	QuARS [22]	[10]	RQA [28]	SREE [70]	Smella [24]	RETA [4]
Ambiguous Adv. & Adj.							E/Q	
Comparatives							E/Q	
Loopholes (or Options)	Q	E/Q	E/Q	E	E	Q*/P*	E/Q	
Negative Terms						O	E/Q	
Non-Verifiable Terms							E/Q	
Pronouns						O	E/Q	O
Subjectivity		E/Q	E/Q	E	E	Q*/P*	E/Q	
Superlatives							E/Q	
							E/Q	

Legend: O=No empirical analysis, E=Examples from Case, Q=Quantification, P=Precision analyzed, R=Recall analyzed, * = Aggregated over multiple smells

an evaluation, a quality definition and a technical perspective.

First, one gap in existing automatic QA approaches is the lack of empirical evidence, especially under realistic conditions. Only few of the introduced contributions were evaluated using industrial requirements artifacts. Those who do apply their approach on such artifacts focus on quantitative summaries explaining which finding was detected and how often it was detected. Some authors also give examples of findings, but only few works analyze this aspect in depth with precision and recall, especially in the fuzzy domain of ambiguity (see Table 2). When looking at the characteristics that are described in ISO 29148, we have not seen a quantitative analysis of precision and recall. Furthermore, reported evidence does not include qualitative feedback from engineers who are supposed to use the approach, which could reveal many insights that cannot be captured by numbers alone. However, we postulate that the accuracy of quality violations very much depends on the respective context. This is especially true for the fuzzy domain of natural language where it is important to understand the (context-specific) impact of a finding to rate its detection for appropriateness and eventually justify resolving the issue.

Second, the existing approaches are based on proprietary definitions of quality, based on experience or, sometimes, simply on what can be directly measured. The ARM tool [74] is loosely based on the

IEEE 830 [32] standard. However, as the recent literature survey by Schneider and Berenbach [67] states: “*the ISO/IEC/IEEE 29148:2011 is actually the standard that every requirements engineer should be familiar with*”. We are not aware of an approach that evaluates the current ISO 29148 standard [33] in this respect. As Table 1 shows, for most language quality defects of ISO 29148, there has not yet been a tool to detect these quality defects. To all our knowledge, for neither of these factors, there is an differentiated empirical analysis of precision and recall. Yet, many other quality models (most notably from the ambiguity handbook by Berry et al. [7]) and quality violations could lead to Requirements Smells, as far as they comply with the definition given in the next section.

Finally, taking a more technical perspective, our Requirements Smell detection approach does not fundamentally differ from existing approaches. Similar to previous works, we apply existing NLP techniques, such as lemmatization and POS tagging, as well as dictionaries. For the rules of the ISO 29148 standard, no parsing or ontologies (as used in other approaches) were required. However, to detect superlatives and comparatives in German, we added a morphological analysis, which have not yet seen in related work.

In summary, in our contribution, we extend the current state of reported evidence on automatic QA for requirements artifacts via systematic studies in terms of distribution, precision, recall, and relevance, as well as by means of a systematic evaluation with

Table 2: Related approaches and tools, and their evaluation, detailed supplementary material can be found online [25]

Tool/ Approach	Purpose (unless ambiguity det.)	Publications	Evaluation	Precision	Recall
ConQAT (Palessi)	Redundancy	[34] [21] [62]	E/Q/P Q/P/R Q/P/R	0.27-1 up to 96 0.63	— up to 96 0.86
ReqAlign	Redundancy				
RETA	Structured Language Rules	[4]	E/Q/P/R	0.85-0.94	0.91-1
AQUSA	User Story Rules	[51]	E/Q/P	0.63-1	—
CIRCE	Structured Language Rules	[29] [1]	E	—	—
(Ciemniewska)		[12]	E	—	—
(Kof)		[44]	E/Q	—	—
(Kiyavitskaya)		[40]	E/Q	—	—
RESI		[46] [47] [48]	E/Q	—	—
HeRA		[41] [42]	E	—	—
Alpino		[16]	E/Q	—	—
(Chantree)		[11]	E/P/R	0.6-1	0.02-0.58
Gleich		[30]	E/Q*/P*/R*	0.34-0.97	0.53-0.86
(Krisch)		[49]	E/Q/P	0.12	—
ARM	RE Artifact Metrics	[74]	Q	—	—
QuARS / SyTwo		[19] [18] [22] [10]	E/Q	—	—
RQA		[28]	O	—	—
SREE		[70]	Q*/P*	0.66-0.68*	—
Smella		[24]	E/Q	—	—

Legend: O=No empirical analysis, E=Examples from Case, Q=Quantification, P=Precision analyzed, R=Recall analyzed,
^{*}=Aggregated over multiple smells

practitioners under realistic conditions. We perform this on both existing, as well as new quality defects taken from the ISO 29148. Therefore, we extend our previously published first empirical steps [24] to close these gaps by thorough empirical evaluation.

3. Requirements Smells

We first introduce the terminology on Requirements Smells as used in this paper. In a second step, we define those smells we derived from ISO 29148 and which we use in our studies, before describing the tool realization in the next section.

3.1. Requirements Smell terminology

Code smells are supposed to be an imprecise indication for bad code quality [27]. We apply this concept of smells to requirements and define it as follows: A Requirements Smell is an indicator of a quality violation, which may lead to a defect, with a concrete location and a concrete detection mechanism. In detail, we consider a smell as having the following characteristics:

1. A Requirements Smell is an *indicator* for a quality violation of a requirements artifact. For this definition, we understand requirements quality in terms of quality-in-use, meaning that bad requirements artifact quality is defined by its (potential) negative effects on activities in the software life-cycle that rely on these requirements artifacts (see also [26]).
2. A Requirements Smell does *not necessarily lead to a defect* and, thus, has to be judged by the context (supported e.g. by (counter-/))indications). Whether a Requirements Smell finding is or is not a problem in a certain context must be individually decided for that context and is subject to reviews and other follow-up quality assurance activities.
3. A Requirements Smell has a *concrete location* in an entity of the requirements artifact itself, e.g. a word or a sequence. Requirements Smells always provide a pointer to a certain location that

QA must inspect. In this regard, it differs from general quality characteristics, e.g. completeness, that only provide abstract criteria.

4. A Requirements Smell has a *concrete detection mechanism*. Due to its concrete nature, Requirements Smells offer techniques for detection of the smells. These techniques can, of course, be more or less accurate.

Furthermore, we define a *quality defect* as a concrete instance or manifestation of a quality violation in the artifact, in contrast to a *finding* which is an instance of a smell. However, like a smell indicates for a quality violation, the finding indicates for a defect. Fig. 1 visualizes the relation of these terms.

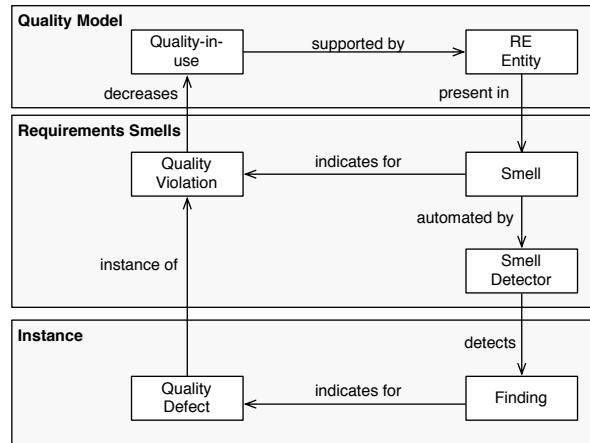


Figure 1: Terminology of Requirements Smells (simplified)

In the following, we will focus on natural language Requirements Smells, since requirements are mostly written in natural language [58]. Furthermore, the real benefits of smell detection in practice should come with automation. Therefore, the remainder of the paper discusses only Requirements Smells where the detection mechanism can be executed automatically (i.e. it requires no manual creation of intermediate or supporting artifacts).

3.2. Requirements Smells based on ISO 29148

We develop a set of Requirements Smells based on an existing definition of quality. For the investigations in scope of this paper, we take the ISO 29148 requirements engineering standard [33] as a baseline. The reasons for this are two-fold.

First, the ISO 29148 standard was created to harmonize a set of existing standards, including the IEEE 830:1998 [32] standard. It differentiates between quality characteristics for a set of requirements, such as completeness or consistency, and quality characteristics for individual requirements, such as unambiguity and singularity. The standard furthermore describes the usage of requirements in different project phases and gives exemplary contents and structure for requirements artifacts. Therefore, we argue that this standard is based on a broad agreement and acceptance. Recent literature studies come to the same conclusion [67].

Second, the standard provides readers with a list of so-called *requirements language criteria* which support the choice of proper language for requirements artifacts. The authors of the standard argue that violating the criteria results "*in requirements that are often difficult or even impossible to verify or may allow for multiple interpretations*" [33, p.12]. For defining our smells, which we describe next, we refer to this section of the standard and use all the defined requirements language criteria. We employ those criteria as a starting point and define the smells by adding the affected entities (e.g. a word) and an explanation. Here, we do not discuss the impact smells have on the quality-in-use. Essentially, smells hinder the understandability of requirements and consequently their subsequent handling and their verification (for a richer discussion, see also previous work in [26]).

Our current understanding is based on the examples given by the standard. A subset of the language criteria, namely **Subjective Language**, **Ambiguous Adverbs and Adjectives** and **Non-verifiable Terms**, as defined in [33], are strongly related, essentially since subjective language is a special type of ambiguity, which may lead to issues during verification. Since the intention of this work is to start with the standard as a definition of

quality, in the following, we will remain with the provided definition based on the language criteria and leave the development of a precise and complete set of Requirements Smells to future work. In detail, we use the requirements language criteria to derive the smells summarized next.

Smell Name:	Subjective Language
Entity:	Word
Explanation:	Subjective Language refers to words of which the semantics is not objectively defined, such as <i>user friendly</i> , <i>easy to use</i> , <i>cost effective</i> .
Example:	The architecture as well as the programming must ensure a simple and efficient maintainability.

Smell Name:	Ambiguous Adverbs and Adjectives
Entity:	Adverb, Adjective
Explanation:	Ambiguous Adverbs and Adjectives refer to certain adverbs and adjectives that are unspecific by nature, such as <i>almost always</i> , <i>significant</i> and <i>minimal</i> .
Example:	If the (...) quality is too low , a fault must be written to the error memory.

Smell Name:	Loopholes
Entity:	Word
Explanation:	Loopholes refer to phrases that express that the following requirement must be fulfilled only to a certain, imprecisely defined extent.
Example:	As far as possible , inputs are checked for plausibility.

Smell Name:	Open-ended, Non-verifiable Terms	Smell Name:	Vague Pronouns
Entity:	Word	Entity:	Pronoun
Explanation:	Open-ended, non-verifiable terms are hard to verify as they offer a choice of possibilities, e.g. for the developers.	Explanation:	Vague Pronouns are unclear relations of a pronoun.
Example:	The system may only be activated, if all required sensors (...) work with sufficient measurement accuracy.	Example:	The software must implement services for applications, which must communicate with controller applications deployed on other controllers.
Smell Name:	Superlatives	Smell Name:	Incomplete References
Entity:	Adverb, Adjective	Entity:	Text reference
Explanation:	Superlatives refer to requirements that express a relation of the system to all other systems.	Explanation:	Incomplete References are references that a reader cannot follow (e.g. no location provided).
Example:	The system must provide the signal in the highest resolution that is desired by the signal customer.	Example:	[1] "Unknown white paper". Peter Miller.
Smell Name:	Comparatives	4. Smella: A prototype for Requirements Smell detection	
Entity:	Adverb, Adjective	Requirements Smell detection, as presented in this paper, serves to support manual quality assurance tasks (see also the next section). The smell detection is implemented on top of the software quality analysis toolkit ConQAT, ⁴ a platform for source code analysis, which we extended with the required NLP features. In the following, we introduce the process for the automatic part of the approach, i.e. the detection and reporting of Requirements Smells. To the best of our knowledge, there is no tool, other than the ones mentioned in related work, that detect and present these smells in natural language requirements documents.	
Explanation:	Comparatives are used in requirements that express a relation of the system to specific other systems or previous situations.	The process takes requirements artifacts in various formats (MS Word, MS Excel, PDF, plain text, comma-separated values) and consists of four steps (see also Fig. 2):	
Example:	The display (...) contains the fields A, B, and C, as well as more exact build infos.	1. <i>Requirements parsing</i> of the requirements artifacts into single items (e.g. sections or rows), resulting in plain texts, one for each item	
Smell Name:	Negative Statements	<hr/> ⁴ http://www.conqat.org	
Entity:	Word		
Explanation:	Negative Statements are "statements of system capability not to be provided"[33]. Some argue that negative statements can lead to underspecification, such as lack of explaining the system's reaction on such a case.		
Example:	The system must not sign off users due to timeouts.		

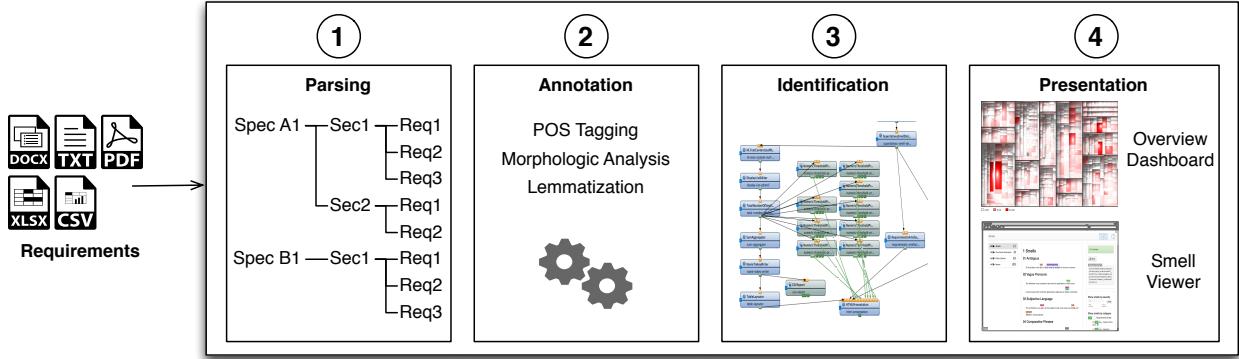


Figure 2: The overall smell detection process

2. *Language annotation* of the requirements with meta-information
3. *Findings identification* in the requirements, based on the language annotations
4. *Presentation* of a human-readable visualization of the findings as well as a summary of the results

The techniques behind these steps are explained in the following section.

4.1. Requirements parsing

Our current tool is able to process several file formats: MS Word, MS Excel, PDF, plain text and comma-separated values (CSV). Depending on the format, the files are parsed in different ways. Plain text and PDF are taken as is and parsed file by file. Microsoft Word files are grouped by their sections. For Microsoft Excel and CSV files, we define those columns that represent the IDs or names (if there are any), and those columns should be used as text input to detect smells.

If a file is written in a known template, such as a common template for use cases, we can make use of this template to understand structural defects, such as lacking content items in a template. In the remainder of this paper, however, we focus on the natural language Requirements Smells as provided by the ISO standard.

4.2. Language annotation

For the annotation and smell detection steps, we employ three techniques from Natural Language Processing (NLP) [36]. Table 3 additionally shows which of the techniques we use for which smell.

POS Tagging: For two smells, we use part-of-speech (POS) tagging. Given a sentence in natural language, it determines the role and function of each single word in the sentence. The output is a so-called *tag* for each word indicating, for instance, whether a word is an adjective, a particle, or a possessive pronoun. We used the Stanford NLP library [71] and the RFTagger [66] for this. Both are statistical, probabilistic taggers that train models similar to Hidden Markov Models based on existing databases of tagged texts. A detailed introduction into the technical details of POS tagging is beyond the scope of this paper but can be found, for example, in [36]. We use POS tagging to determine so-called substituting pronouns. These are pronouns that do not repeat the original noun and, thus, need a human's interpretation of its dependency.

Morphological Analysis: Based on POS tagging, we perform a more detailed analysis of text and determine a word's inflection. This includes, inter alia, determining a verb's tense or an adjective's comparison. We use this technique to analyze if

adjectives or adverbs are used in their comparative or superlative form.

Dictionaries & Lemmatization: For the remaining five smells, we use dictionaries based on the proposals of the standard [33] and on our experiences from first experiments in a previous work [24]. We furthermore apply lemmatization for these words, which is a normalization technique that reproduces the original form of a word. In other words, if a lemmatizer is applied to the words *were*, *is* or *are*, the lemmatizer will return for all three the word *be*. Lemmatization is in its purpose very similar to stemming (see, e.g. the Porter Algorithm [61]), yet not based on heuristics but on the POS tag as well as the word’s morphological form. For Requirements Smells, the difference is significant: For example, the words *use* and *useful* stem to the same word origin (*use*), but to different lemmas (i.e. meanings; *use* and *useful*). Whereas the lemma *use* is mostly clear to all stakeholders, the lemma *useful* is easily misinterpreted.

4.3. Findings identification

Based on the aforementioned information, we identify findings. This step actually finds the parts of an artifact that exhibit bad smells. Dependent on the actual smell, we use different techniques, as shown in Table 3. If the smell relates to a grammatical aspect, we search through the information from POS tagging and morphological analyses. For example, for the Superlatives Smell, we report a finding if an adjective is, according to morphologic analysis, inflected in its superlative form. If the smell does not relate to grammatical aspects but rather the semantics of the requirements, we identify the smell by matching the lemma of a word against a set of words from pre-defined dictionaries. Since the requirements under analysis in our cases did not contain references, incomplete references are not part of our tool at present.

4.4. Findings presentation

We implemented the presentation of findings in a prototype, which we call *Smella* (Smell Analysis).

Smella is a web-based tool that enables viewing, reviewing and blacklisting findings as well as a hotspot analysis at an artifact level. In the Smella presentation, we display the complete requirements artifact and annotate findings in a spell checker style. This follows the idea of smells as only indications that must be evaluated in their context. Lastly, the system gives detailed information when a user hovers a finding (see Fig. 3). In the following, we shortly describe the features of Smella in detail to provide the reader with a rough understanding of the prototype.

View findings: At the level of a single artifact, we present the text of the artifact and its structure. We mark all findings in the text. With a click on the markers, more information about the finding is displayed. The tool provides an explanation of the rationale behind this smell and possible improvements for the finding depending on the smell (every smell has a message for improvements).

Review findings: We allow the user to write a review and to set a status for each finding, both supporting feedback mechanisms within and between project teams. A user has the possibility to accept or reject a finding but also to set a custom state, for example *under review*. Accepting a finding means the finding needs to be addressed. If a finding is rejected, the finding does not need to be addressed. The semantics of the custom status is open to the reviewer.

Blacklist findings: Smells are only indicators for issues. Therefore, users can reject findings. If a finding is rejected by the user, the finding is removed from the visualization and will not be presented to the user anymore.

Disable smells: Often, users are interested in only a subset of smells or even just one smell. Therefore, we allow the user to hide all findings of particular smells and to select the smells she wants to display in the artifact view.

Analyze hotspots: In this view, we present all artifacts in a colored treemap (see Fig. 4). Every box in the treemap is one artifact, with the color

The screenshot shows the Smella tool interface. On the left, a sidebar lists cases: Case A, Case B, and Case C (selected). Below Case C is a list of requirements from Requirement01.txt to Requirement14.txt. The main area displays requirement artifacts and their associated smells.

- Requirement01.txt:**
 - As a visitor, I want to see the checkboxes in the different categories displayed more clearly, so that I can see more quickly that I can select and deselect categories.
 - Smell:** Loop Comp
- Requirement02.txt:**
 - As a visitor, I want to see the checkboxes in the different categories displayed more clearly, so that I can see more quickly that I can select and deselect categories.
 - Smell:** Loop Comp
- Requirement03.txt:**
 - As an editor, I want to make it simpler
 - Smell:** Comp
- Requirement04.txt:**
 - As a visitor, I want to see further details
 - Smell:** Comp
- Requirement05.txt:**
 - As a customer, I want to have a larger number of E-Mails in my mailbox
 - Smell:** Comp

On the right, there are two filtering panels:

- Show smells by severity:** A slider from 'less severe' to 'more severe' with an 'Apply' button.
- Show smells by category:** A list of smells with status indicators (ON/OFF) and descriptions:
 - Ambi (OFF): Ambiguous Adverbs and ...
 - Comp (ON): Comparative Requirements ...
 - Loop (ON): Loophole Smell
 - Nega (OFF): Negative Words Smell
 - Non- (OFF): Non-verifiable Term Smell
 - Subj (OFF): Subjective Language Smell
 - Supr (OFF): Superlative Requirements S...
 - Vagu (OFF): Vague Pronouns Smell

Figure 3: A sample output from the smell detection tool (detailed artifact view) with some smells disabled and some findings blacklisted

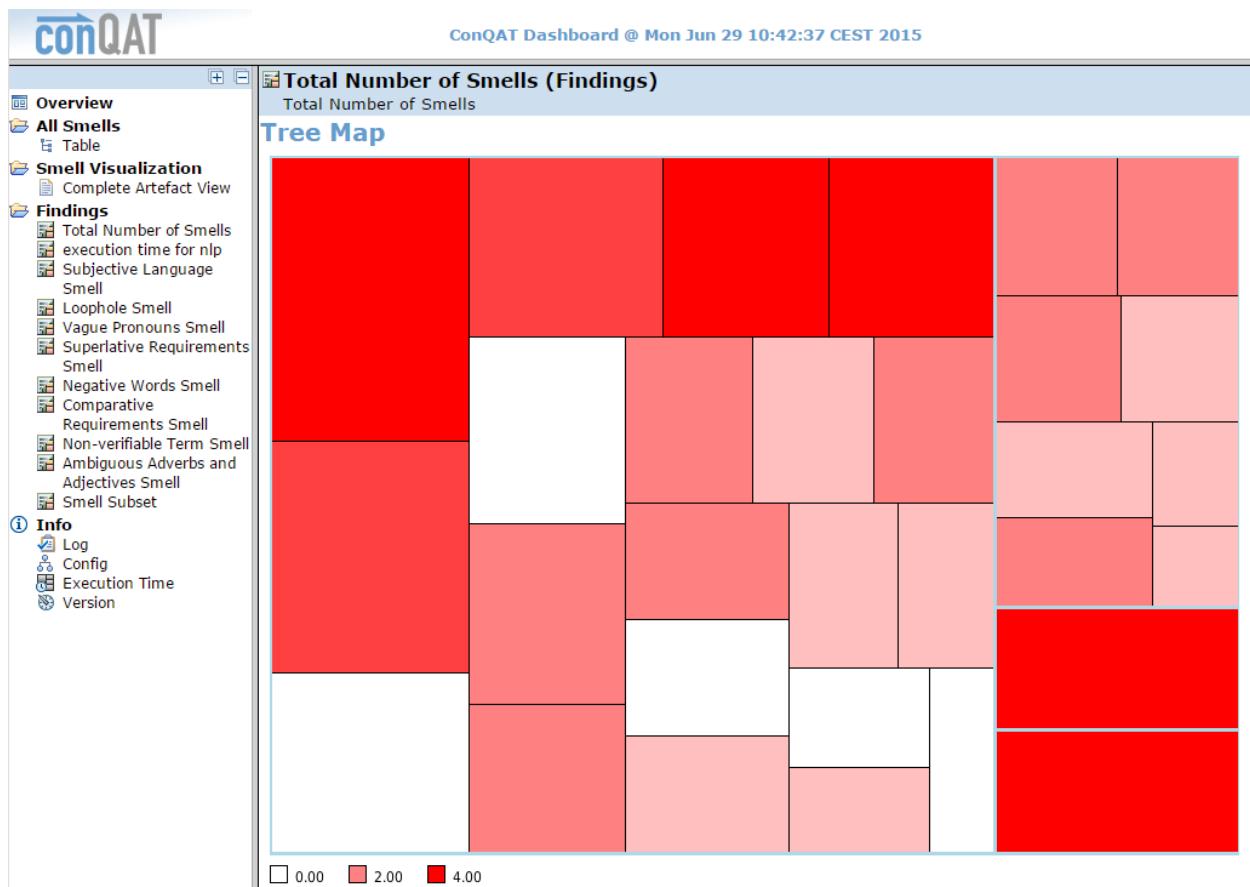


Figure 4: A sample output from the smell detection tool (hotspot analysis view)

Table 3: Detection techniques for smells

Smell Name	Detection Mechanism
Subjective Language	Dictionary
Ambiguous Adverbs and Adjectives	Dictionary
Loopholes	Dictionary
Open-ended, non-verifiable terms	Dictionary
Superlatives	Morphological analysis or POS tagging
Comparatives	Morphological analysis or POS tagging
Negative Statements	POS tagging and dictionary
Vague Pronouns	POS tagging: Substituting pronouns.
Incomplete References	Not in scope of this study

of the box indicating the number of findings: the more red an artifacts is, the more findings it contains (the more it “smells” bad). The artifacts are grouped by their folder structure. The tool provides a summarized treemap for all smells as well as a separate treemap for all individual smells. With these treemaps, users can identify artifacts or groups of artifacts exhibiting a high number of findings – for one single smell but also for all smells together. This feature supports the identification of candidates for in-depth reviews.

5. Requirements Smell detection in the process of quality assurance

The Requirements Smell detection approach described in previous sections serves the primary purpose of supporting quality assurance in RE. The detection process itself is, however, not restricted to particular quality assurance tasks, nor does it depend on a particular (software) process model as we will show in Sect. 6. Hence, a smell detection, similar to the notion of quality itself, always depends on the views in a socio-economic context. Thus, how to integrate smell detection into quality assurance needs to be answered according to the particularities of that context. In the following, we therefore briefly outline the role smell detection can generally take in the process of quality assurance. More concrete proposals on how to integrate it into specific contexts are given in our case studies in Sect. 6.

We postulate the applicability of the Requirements Smell detection in the process of both constructive and analytical quality assurance (see Fig. 5). From the perspective of a constructive quality assurance, authors can use the smell detection to increase their awareness of potential smells in their requirements artifacts and to remove smells before releasing an artifact for, e.g., an inspection. External reviewers in turn, can then use the smell detection to prepare analytical, potentially cost-intensive, quality assurance tasks, such as a Fagan inspection [20]. Such an inspection involves several reviewers and would benefit from making potential smells visible in advance. Iterative inspection approaches are also known as phased inspections, as defined by Knight and Myers [43].

We furthermore believe that one major advantage is that the scope of our smell detection is not to enforce resolving a potential smell but to increase the awareness of the like and to make transparent later reasoning why certain decisions have been taken. Please note that two different roles (e.g. requirements engineer and QA engineer) can take two different viewpoints on the same smell, respectively its criticality and whether it should be resolved or not. In addition, a finding could be unambiguous to the author, but unclear to the target group of readers (represented by the reviewers). Therefore, one contribution of our tool-supported smell detection is also to actively foster the communication between reviewers and authors and to enable continuous feedback between both roles. For this reason, we enable stakeholders in Smella to com-

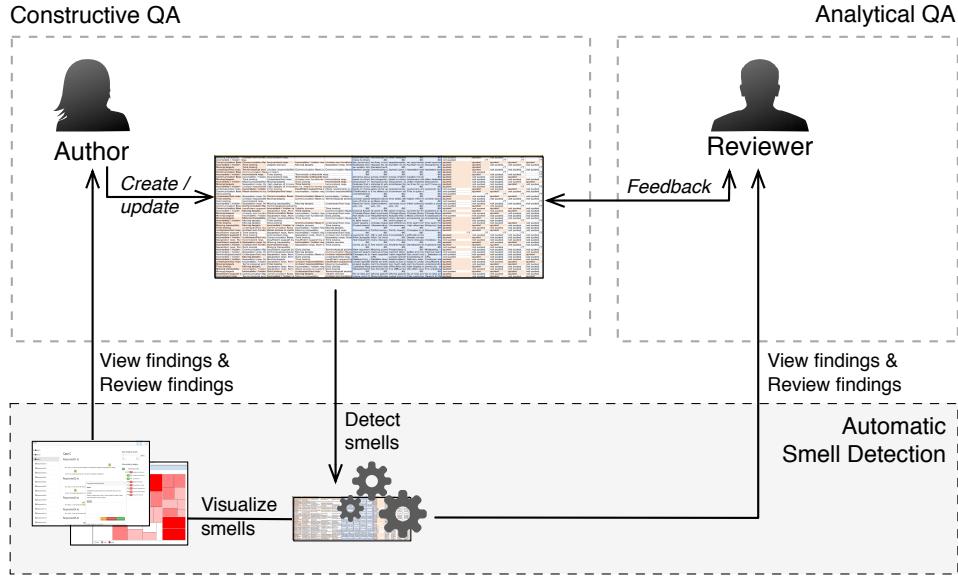


Figure 5: A suggestion for applying Requirements Smell detection in QA

ment on detected smells and make explicit whether they need to be resolved or whether and why they have been accepted or rejected.

6. Evaluation

For a better, empirical understanding of smells in requirements artifacts, we conducted an exploratory multi-case study with both industrial and academic cases. We particularly rely on case study research over other techniques, such as controlled experiments, because we want to evaluate our approach in practical settings under realistic conditions. For the design and reporting of the case study, we largely follow the guidelines of Runeson and Höst [63].

6.1. Case study design

Our overall research objective is as follows:

Research Objective: Analyze whether automatic analysis of Requirements Smells helps in requirements

artifact quality assurance.

To reach this aim, we formulate four research questions (RQ). In the following, we introduce those research questions, the procedures for the case and subjects selection, the data collection and analysis, and the validity procedures.

6.1.1. Research questions

RQ 1: How many smells are present in requirements artifacts? To see if the automatic detection of smells in requirements artifacts could help in QA, we first need to verify that Requirements Smells exist in the real world. The answer to this question fosters the understanding how widespread the smells under analysis are in industrial and academic requirements artifacts.

RQ 2: How many of these smells are relevant? Not only the number of detected smells is important. If many of the detected smells are false positives and not relevant for the requirements engineers and developers, it would hinder QA more than it would help.

As relevancy is a rather broad concept, we break down RQ 2 into two sub-questions.

RQ 2.1: How accurate is the smell detection? The first sub-question looks at the more technical view on relevance. We want to find false positives and false negatives to determine the precision and recall of the analysis in terms of correct detection of the defined smell.

RQ 2.2: Which of these smells are practically relevant in which context? This second sub-question is concerned with practical relevance. We investigate whether practitioners would react and change the requirement when confronted with the findings.

RQ 3: Which requirements quality defects can be detected with smells? After we understood how relevant the analyzed Requirements Smells are, we want to understand their relation to existing quality defects in requirements artifacts. Hence, we need to check whether, and if so, which defects in requirements artifacts correspond to smells, as we understand smell findings as indicators for defects.

RQ 4: How could smells help in the QA process? Finally, we collect general feedback from practitioners whether (and how) smell detection could be a useful addition to QA for requirements artifacts and whether as well as how they would integrate the smell detection into their QA process.

6.1.2. Case and subjects selection

Our case and subject selection is opportunistic but in a way that maximizes variation and, hence, evaluates the smell detection in very different contexts. This is particularly important for investigating requirements artifacts under realistic conditions, also due to the large variation in how these artifacts manifest themselves in practice. A prerequisite for our selection is the access to the necessary data. To get a reasonable quantitative analysis of the number of smells (RQ 1) and qualitative analysis of the relation of smells and defects (RQ 3), we complement our three industrial cases with a case in an academic setting. There, various student teams are asked to provide software with a certain set of (identical) functionality for a customer as part of a practical course. This is also a realistic setting but provides us with a

higher number of specifications and reviews than in the industrial cases.

We will refer to the subjects of the industrial cases as *practitioners* and we will call the latter subjects *students*.

6.1.3. Data collection procedure

We used a 6-step procedure to collect the data necessary for answering the research questions.

1. *Collect requirements artifact(s) for each case.* We retrieved the requirements artifacts to be analyzed in each case. For one case, the requirements were stored in Microsoft Word Documents. For the other cases, this involved extracting the requirements from other systems, either a proprietary requirements management tool (resulting in a list of html files), or the online task management system JIRA, which led to a set of comma-separated values files. For the student projects, the students handed in their final artifacts either as a single PDF or as a PDF with the general artifact and another PDF with the use cases. Where authors explicitly structured requirements in numbered requirements, user stories or use cases, we counted these artifacts.
2. *Run the smell detection via Smella.* We applied our detection tool as introduced in Sect. 4.4 on the given requirements artifacts, which generated a list of smells per artifact.
3. *Classify false positives.* For all cases in which we wanted to present our results to practitioners, we reviewed each detected finding. In pairs of researchers, we classified the findings as either true or false positive. We classified a finding as false positive if the finding was not an instance of the smell, e.g. because the results of the linguistic analysis was incorrect.⁵ For artifacts containing more than 10 findings of a smell, we only inspected a set of 10 random findings (of that

⁵For example, if the linguistic analysis incorrectly classified the word *provider* in the sentence “*As a provider, I want [...]*” as a comparative adjective.

smell) per artifact. The same holds for Case D, where we inspected 10 random findings of each category for the whole case.

4. *Inspect documents for false negatives.* To calculate the recall of the smell detection, for each case we randomly selected one artifact that a pair of researchers inspected for false negatives. To ease the manual inspection, we grouped the smells **Subjective Language**, **Ambiguous Adverbs and Adjectives**, **Loopholes**, **Non-verifiable Terms** (as **Ambiguity-related smells**). We classified whether a finding is a true or false negative based on the same conditions as in the previous step.

One common cause for false negatives for dictionary-based smells can be that an ambiguous phrase is not part of the dictionary. Since we developed the dictionaries based on existing dictionaries, such as the standard, these dictionaries are not yet complete and must be further developed. However, since this is an issue that is not a problem of the smell detection approach in general, but rather a configuration task, we did not take these findings into consideration for the recall.

5. *Get rating by practitioners.* We selected a subset of the true positive findings so that we cover all smells with a minimum of two findings per smell as far as the artifacts allowed. When we found repeating or similar findings, e.g. multiple similar sentences with the same smell, we also included one of these findings into the set.

We presented this subset to the practitioners and interviewed them, finding by finding, through three closed questions (see also Table 9): Q1: Would you consider this smell as relevant? Q2: Have you been aware of this finding before? Q3: Would you resolve the finding? Of these, the former two must be answered with *yes* or *no*. For the last question, we also needed to take the criticality into account. Therefore, in case practitioners answered that they would resolve a finding, we also asked whether they would resolve it immediately, in a short time (i.e. within this

project iteration) or in a long time (e.g. if it happens again). In addition to these three questions, we took notes of qualitative feedback, such as discussions.

6. *Interview practitioners.* In addition to the ratings, we performed open interviews with practitioners about their experience with the smell detection and how they might include it in their quality assurance process. We took notes of the answers.
7. *Get review results from students.* Lastly, the students performed reviews of the artifacts of other student teams. They documented and classified found problems according to a checklist (see Table A.11) without awareness of the smell findings in their artifacts. We then collected the review reports from the students.

6.1.4. Analysis procedure

We structure our analysis procedure into seven steps. Each step leads to the results necessary for answering one of our research questions.

1. *Calculate ratios of findings per artifact.* To understand whether smells are a common issue in requirements artifacts, we compared the quantitative summaries of smells in the various artifacts and domains. To enable a comparison between different types of requirement artifacts, we used the number of words in each artifact as a measure of size. Hence, we finally reported the ratio of findings per 1000 words for each smell and all smells in total. This provided answers for RQ 1.
2. *Calculate ratios of findings for parts of user stories.* In one case, we had a common structure of the requirements, because they were formulated as user stories. To get a deeper insight into the distribution of smells and findings, we calculated the ratios of findings per 1000 words for each part. We divided the user stories into the parts *role* (“As a...”), *feature* (“I want to...”) and *reason* (“so that...”) using regular expressions. We counted the words and findings in each part. This provided further insights into the answer for RQ 1.

3. *Calculate ratios of false positives.* After a rough overview obtained under the umbrella of RQ 1 describing the number of findings for each smell of the varying artifacts, we wanted to better understand the smell's relevance. The first step was to calculate the ratios of false positive as we classified them in Step 3 of the data collection. We reported false positive rates overall and for each smell. This provides the first part of the answer to RQ 2.1.
4. *Calculate ratios of false negatives.* The precision of a smell detection is tightly coupled with the recall. Therefore, we calculated the ratio of detected smell findings to all existing findings, according to our manual inspection, as described in Step 4 of the data collection procedure. This provides the second part of the answer to RQ 2.1.
5. *Calculate ratio of irrelevant smells.* We were not only interested in errors in the linguistic analysis but also in how relevant the correct analyses were for the practitioners. Hence, we calculated and reported the ratios of findings considered irrelevant by the practitioners. This answers RQ 2.2.
6. *Compare defects from reviews with findings.* From the students, we received review reports for each artifact. As the effort to check them all would have been overwhelming, we took a random sample of 20% of the artifacts. For each of the defects detected in the review, we checked if there is a corresponding finding from a smell. This answers RQ 3.
7. *Interpret interview notes.* To answer finally RQ 4, we analyze the interview transcripts and code the answers given by the interviewees manually.

6.1.5. Validity procedure

First, we used peer debriefing in the sense that all data collection and analyses were done by at least two researchers. Analysis results were also checked by all researchers. This researcher triangulation especially increases the internal validity. Furthermore, we kept

an audit trail in a Subversion system to capture all changes to documents and analyses.

Second, we performed all the classifications of findings into true and false positives in pairs. This already helped to avoid misclassifications. To further check our classifications, we afterwards did an independent re-classification of randomly selected 10% of the findings and calculated the inter-rater agreement. We discussed to clarify which findings we consider false positives and repeated the classifications until we reached an acceptable agreement. The same procedure held for the inspection of artifacts to detect false negatives, which we also conducted in pairs. Furthermore, we also independently re-classified one of the artifacts to understand the inter-rater agreement on the false negatives. Overall, our analysis for false positives and relevance of the findings is also a validity procedure in the sense that we check in RQ 2 the results from RQ 1.

Third, we discussed with the practitioners what relevance of smells means in the context of the study to avoid misinterpretations. Furthermore, we gave the students review guidelines to give them an indication what quality defects in requirements artifacts might be. Both serve in particular as mitigation to threats to the internal and the construct validity.

Fourth, we performed the analysis of the correspondence between smells and defects with a pair of researchers. This pair derived a classification of the found and not found defects. Both other researchers reviewed the classification, and we improved it iteratively until we reached a joint agreement.

Fifth, we performed member checking by showing our transcriptions and interpretations for RQ 4 to the interviewed practitioners and incorporating feedback.

Finally, to support the external validity of the results of our study, we aimed at selecting cases with maximum variation in their domains, sizes, and how they document requirements.

6.2. Results

In the following, we report on the results of our case studies. We first describe the cases and subjects under analysis, before we answer the research questions. We end by evaluating the validity of the cases.

6.2.1. Case and subjects description

The first three cases contain requirements produced in different industrial contexts: embedded systems in the automotive industry, business information systems for the chemical domain and agile development of web-based systems. While the first two represent rather classical approaches to Requirements Engineering, the third case applies the concept of user stories, as it is popular in agile software development. The fourth case is in an academic background and employs both use cases and textual requirements. Regarding subject selection, for each industrial case we selected practitioners involved in the company, domain and specification. We executed the findings rating (Step 5) and the interviews regarding the QA process (Step 6) with the same experts, so that their answer in Step 6 is based on their experience with practical, real examples. In the following, we describe the cases, as well as the experts or students for each case. Table 4 provides a quantitative overview of the cases.

Case A: Daimler AG. Daimler AG is a multinational automotive corporation headquartered in Stuttgart, Germany. At Daimler, we analyzed six different requirements artifacts (A1–A6) which were written by various authors. The requirements artifacts describe functionality in different domains of engine control as well as driving information. In this case, requirements are written down in the form of sentences, identified by an ID. The authors are domain experts who are coached on writing requirements.

The requirements artifacts A1–A6 consist of 323 requirements in total (see Table 4). All of the artifacts of Daimler analyzed in our study were created by domain experts in a pilot phase after a change in the requirements engineering process as part of a software process improvement endeavour. For RQ 2.2., we reviewed 22 findings with an external coach who works as a consultant for requirements engineering and has tightly collaborated with the group for many years.

Case B: Wacker Chemie AG. In the second case, we analyzed requirements artifacts of business information systems from Wacker Chemie AG. Wacker is a globally active company working in the chemical sector and headquartered in Munich, Germany. The

systems that we analyzed fulfil company-internal purposes, such as systems for access to Wacker buildings or support systems for document management.

We analyzed three Wacker requirements artifacts that were written by five different authors. At Wacker, functional requirements are written as use cases (including fields for *Name*, *Description*, *Role* and *Pre-condition*) whereas non-functional requirements are described in simple sentences. The artifacts consisted of 53 use cases and 13 numbered requirements (see Table 4). For the reviews of the findings in RQ 2.2, we selected 18 findings and discussed them with the Chief Software Architect, who also has several years of experience in quality assurance.

Case C: TechDivision. For the third case, we analyzed the requirements of the agile software engineering company TechDivision GmbH. TechDivision has around 70 employees, working in 3 locations in Germany. They focus mainly on web development, i.e. creating product portals and e-commerce solutions for a variety of companies, as well as web consulting, especially focusing on search engine optimizations. Many of their products involve customisation of Magento⁶ or Typo3⁷ frameworks.

In their projects, TechDivision follows an agile software development process using either Scrum [68] or Kanban [3] methodologies. For their requirements, TechDivision applies user stories [14], which they write and manage in Atlassian JIRA⁸. User stories at TechDivision follow the common Connextra format: *As a [Role], I want [Feature], so that [Reason]*. We will also follow this terminology here.

The systems under analysis consist of two online shopping portals, a customer-relationship system and a content-management system, all of which we cannot name for non-disclosure-agreement reasons. In total, we analyzed over 1,000 user stories containing roughly 28,000 words. For RQ 2.2, we met with an experienced Scrum Master and a long-term developer, who have worked on several projects for TechDivision.

⁶<http://www.magento.com>

⁷<http://www.typo3.org>

⁸<https://atlassian.com/software/jira>

Case D: University of Stuttgart. The requirements of Case D were created by 52 groups of three 2nd-year students each during a compulsory practical course in the software engineering programme at the University of Stuttgart. We removed one artifact, because it was incorrectly encoded, thus resulting in 51 requirements artifacts for this analysis.

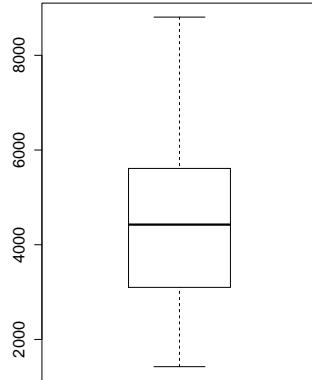


Figure 6: Variation of size of requirements artifacts in Case D in words

The resulting requirements artifacts differ vastly in style; hence, we were unable to count them in terms of requirements, but instead only counted the structured use cases as provided by the authors, and quantified the artifacts by word size. The average size of a requirements artifact was 4,471 words (min: 1,425, max: 8,807, see Fig. 6) and contained 19 use cases (min: 6, max: 39), thus creating a set of artifacts of nearly a quarter of a million words, including more than 950 use cases.

For practical reasons, we could not evaluate each research question in each case: For example, RQ 3 depends on the existence of reviews with documented results, which is often not existent in practice. Furthermore, depending the answers of RQ 4 on the potentially less experienced students from Case D would introduce a threat to the validity of our evaluation.

Table 5 shows the mapping between research questions and study objects. The interviews for RQ 2.2 and RQ 4 lasted 60 minutes for each Case A and B and 120 minutes for Case C.

Table 5: Study objects usage in research questions

Case	RQ 1: Distribution	RQ 2.1: Precision	RQ 2.1: Recall	RQ 2.2: Relevance	RQ 3: Defect Types	RQ 4: QA Process
A: Daimler	✓	✓	✓			✓
B: Wacker	✓	✓	✓			✓
C: TechDivision	✓	✓	✓	✓		✓
D: Univ. of Stuttgart	✓	✓	✓		✓	

6.2.2. RQ 1: How many Requirements Smells are present in the artifacts?

Under this research question, we quantify the number of findings that appear in requirements. Table 6 shows the number of findings for each case, each requirements artifact and each smell and also puts these numbers in relation to the size of the artifact. We analyzed requirements of the size of more than 250k words, on which the smell detection produced in total more than 11k findings, thus revealing roughly 44 findings per thousand words.

Table 6 shows that all requirements artifacts contain findings of Requirements Smells. They vary from 5 findings for the smallest⁹ case (A3) up to 572 for the largest case (C4). The number of findings strongly correlates with the size of the artifact (see Fig. 7, Spearman correlation of 0.9). Hence, in the remainder, we normalize the number of findings by the size of the artifact.

The artifacts of Daimler have an average of 26 findings per thousand words, in contrast to 41 for both

⁹in terms of total number of words

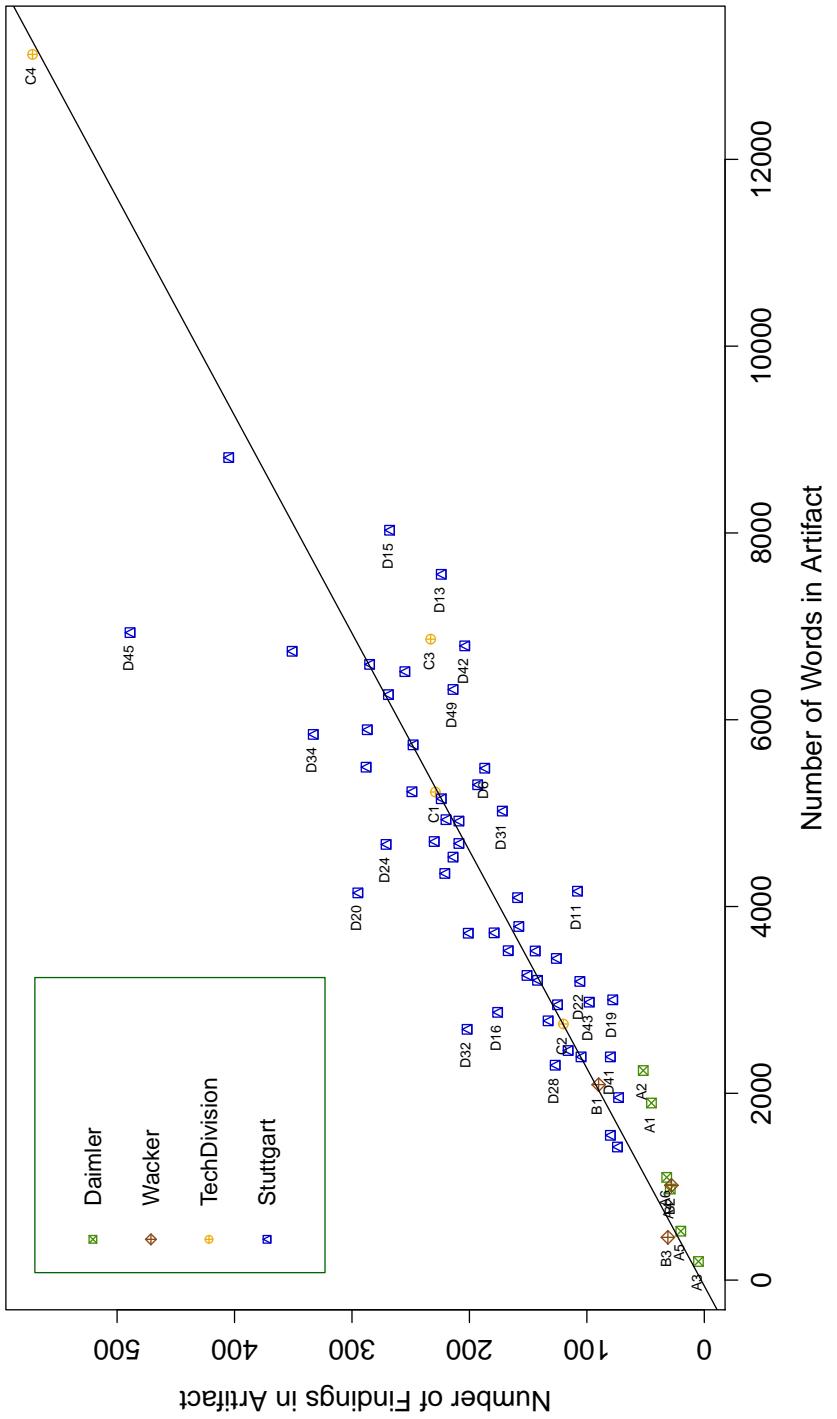


Figure 7: Number of findings strongly correlates with size of artifact (for readability reasons, for the Stuttgart cases (blue) only IDs of less correlating artifacts are displayed).

Wacker and TechDivision and 43 for the artifacts produced by the students. Best to analyze the variance within a requirements artifact seems Case D, in which multiple teams had a similar background and project size. Fig. 8 shows the variance between the artifacts of Case D with an average of 44 findings, a minimum of 26 findings (D11) and a maximum of 75 findings (D32) per 1,000 words.

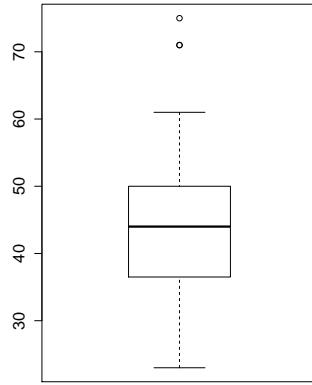


Figure 8: Number of findings per 1,000 words in Case D

When inspecting the different Requirements Smells, we can see that the most common smells are **vague pronouns** with 25 findings per 1,000 words, followed by the **negative words** smell with 6 findings and the **loophole** smell with 4 findings. The least often smells are **non-verifiable terms** with 1 finding per 1,000 words, and **ambiguous adverbs and adjectives** with 0.25 findings per 1,000 words. In fact, the most common smell, **vague pronouns**, appears 100 times more often than the **ambiguous adverbs and adjectives**. To analyze the variance in depth, we again take the students' artifacts for reference. Fig. 9 shows the relative number of findings across the projects.

Interpretation. We interpret the quantitative overview along three variables: projects, contexts and the different Requirements Smells.

Projects When comparing at project level, we see that Cases A1–A6 (with outlier A5) and C1–C4 (with outlier C3) show quite similar numbers. In contrast B1 to B3 vary between 28 and 68 findings per 1,000 words. When looking into the most extreme outliers B3 and D32, we see a systematic error that creates a large number of findings: Both projects repeatedly explain what the system *should*¹⁰ do instead of what it *must* do. 16 of 19 **loopholes** findings in B3 and 29 of 37 **loophole** findings in D32 root from this problem. This can lead to difficult issues in contracting as requirements that are phrased with a *should* are commonly understood as optional (see e.g. RFC2119 [9] for a detailed explanation).

Hence, we could see a surprising consistency in two of three industrial case studies. The Wacker data varies, so does the students case. In both cases, the negative extremes point at issues that potentially have expensive consequences.

Context The four cases differ strongly in their context: They write down requirements in different forms, vary in their software development methodology and also produce software for different domains. When comparing the findings at the domain level, we see that Daimler artifacts with an average of 26 findings per thousand words contain less findings than both Wacker and TechDivision with 41 findings and the artifacts produced by the students with 43 findings.

Our partners reported that there have been trainings for the authors of the cases A1–A6 recently, which could explain the difference. Another reason could be the strong focus that the automotive domain puts on requirements and requirements quality in contrast to the other domains. Lastly, also the strict process in this domain could be a

¹⁰*Soll* is a German modal verb that is less strict than an English *must*.

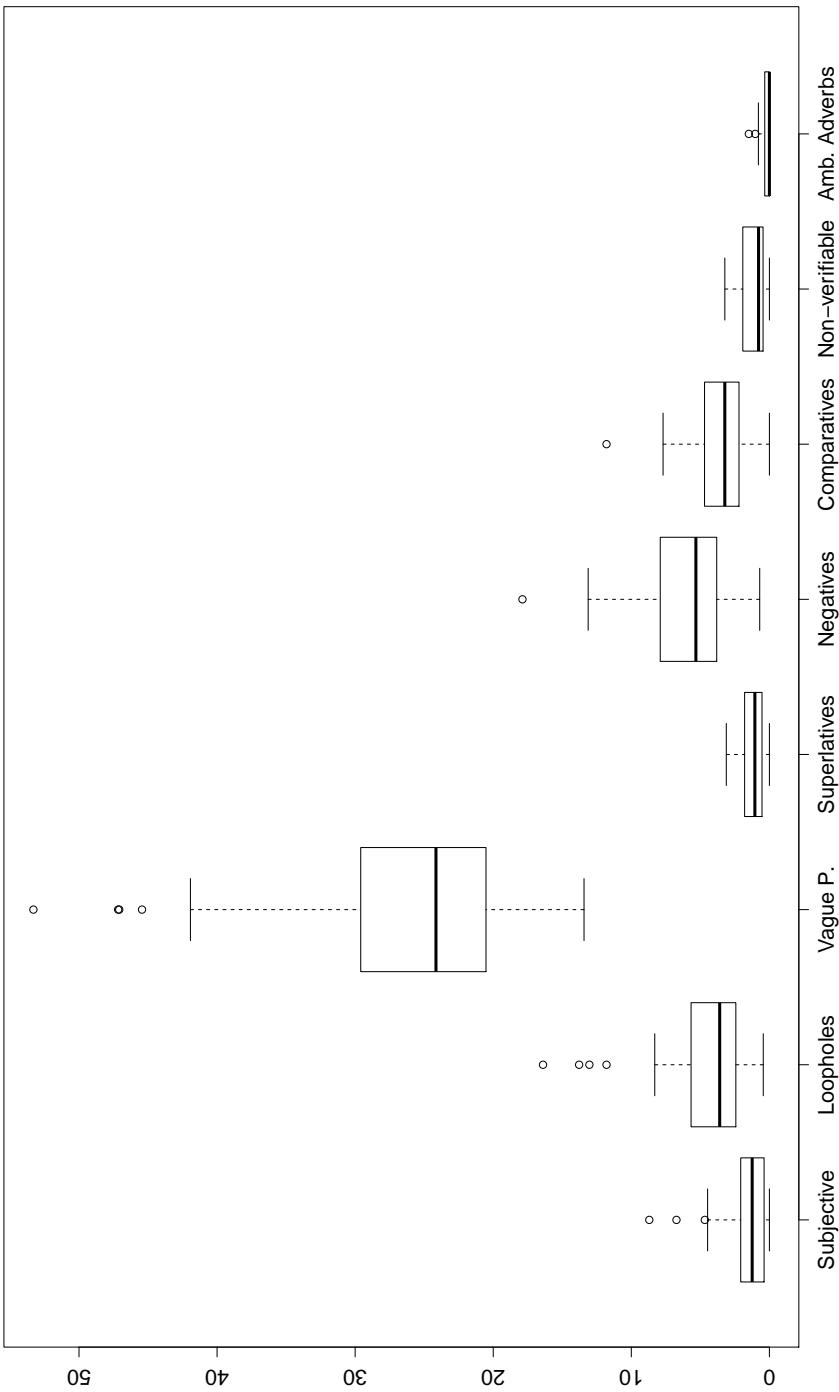


Figure 9: Variation of smells per 1,000 words in Case D

reason for this striking difference of the Daimler requirements. Unsurprisingly, the students' requirements form the lower end of the scale, yet not by much.

Requirements Smells When comparing the eight smells, we see a strong variance between the number of findings, both in absolute as well as relative values. A qualitative inspection indicates reasons for the most occurring smells. First, the smell detection for **vague pronouns** finds all substituting pronouns in the requirements. Especially in German, in many sentences the reference of the pronoun can sometimes be derived from gender and grammatical case of the word, thus correctly detecting pronouns, but not *vague* pronouns. RQ 2.1 quantifies this issue. Second, the most common indication for **loophole** findings is the aforementioned use of the word *should*. We discuss this case in-depth with practitioners in RQ 2.2. Third, we will also inspect reasons for the high number of negative words findings in RQ 2.1 and RQ 2.2.

Answer to RQ 1. The number of findings in requirements artifacts strongly correlates with the size of the artifact. There are roughly 44 findings per 1,000 words and some contexts show a striking similarity in the number of findings for their artifacts. In our cases, the automotive requirements had a lower number of findings whereas student artifacts contained a higher number of findings relative to the size of the artifacts. The most common findings are for the smells **loopholes** and **vague pronouns**.

6.2.3. RQ 2.1: How accurate is the smell detection?

To understand the capabilities of the smell detection, we need to understand precision as metric indicating how many of the detected findings are correct, as well as recall as a metric indicating how many of the correct findings are detected.

Precision. To understand to which extent the numbers of findings for certain smells in RQ 1 are caused by the detection mechanism, we inspected a random sample of 616 findings by taking equivalent sets of

findings from each project and manually classifying whether the finding fulfills the smell definition. We could not inspect the same number of findings of each smell for each project, because some projects only had few or even no findings of a certain smell (see number of findings per project in Table 6).

Table 7 and Fig. 10 show the summary of this analysis: The precision of the detection of the **subjective language** smell revealed only three false positives in total, thus leading to a precision of 0.96. **Non-verifiable words**, **loophole**, and **ambiguous adverbs and adjectives** smells range between 0.70 and 0.81, hence leading to roughly one mistake in four suggestions. **Comparative** and **superlative** smells range around 0.5 which would mean that every second finding is correct. At the rear end of the list are the **negative words** and **vague pronouns** smells with one correct finding in three to four suggestions. Across all smells, the precision is between 0.48 (over all inspections) and 0.59, if we take the varying number of inspected findings between the smells into account. To understand these numbers, we qualitatively inspected the false positive classifications, revealing the following main reasons for false positives:

Grammatical errors in real world language.

The first issue that creates false positives is the fact that our study analyzes real world language. Some of the requirements, especially in Case C, contained a number of grammatical flaws as well as dialectal phrases, which lead to wrong results in the automatic morphologic analysis and automatic POS tagging and consequently also to false positives during smell detection.

Vague pronouns. The smell detection for **vague pronouns** showed the lowest precision. In the detection of this smell, we look for substituting pronouns, which are pronouns where the noun is not repeated after the pronoun¹¹, of which we characterize only every fourth finding as a defect. The reason behind this poor performance, besides a number of false positives due to the poor grammar mentioned before, is the comparably

¹¹E.g. *The father of these.* vs *The father of these kids.*

large number of grammatical exponents of the German language. In addition to number and three grammatical genders, the German language also has four grammatical cases. Therefore, in various instances of substituting pronouns, there is only one grammatical possibility of what the pronoun could refer to.

Findings in conditions. A third reason for false positives is that the smell detection, so far, takes very little context into account. For example, the **comparatives** smell aims at detecting requirements that define properties of the system relative to other systems or circumstances¹². When searching for grammatical comparatives in requirements, roughly 48% of the cases are of the aforementioned kind. In roughly the same number of cases, however, the comparative describes a condition. For example, if the requirement states that *if the system takes more than 1 second to respond [...]*, the comparison is not against another system or circumstance but against absolute numbers. Therefore, in this case, the comparative does not indicate a problem (one could even argue that this is an indicator for *good* quality).

A similar problem holds for the **negative phrases** smell: The smell detection aims at revealing statements of what the system should not do. Often, however, the negative is mentioned in conditions. For example, if the requirements express what to do *if the user input is not zero [...]*, the negation relates to a condition and not to a property of the system.

Recall. When analyzing the accuracy of an automatic detection, we must look not only at precision, but also at recall, i.e. the ratio of all detected findings to all defects of a certain type in an artifact. To this end, we inspected one artifact of each case, in total a

¹²As discussed in Sect. 3.2, the problem of comparatives in requirements is validation: How can we understand whether a system fulfills a requirements if that requirement is stated in a relative instead of an absolute way? What if the system in comparison changes its properties, would this render the requirement suddenly unfulfilled?

set of roughly 16,200 words, and manually identified the findings in each artifact. Due to the problems of distinguishing the various ambiguity-related smells, we analyzed the recall of these four smells as if it was one smell, without further differentiation (see Section 6.1.3).

The manual inspection revealed 200 findings in this artifact sample and an average recall of 0.82. Table 8 and Fig. 10 show the summary of the results: The comparison shows a recall between 0.84 and 0.95 for four of the five investigated smells. The highest recall was achieved by the **Comparative Requirements Smell**, with 0.95, which means that the smell detection missed one in 20 findings. The fifth smell, with the lowest recall, is **Superlative Requirements Smell** with a recall of 0.5. However, this smell is one of the rarest of the smells, as one can also see in the results to RQ 1. Therefore our analysis of the recall of this smell is based on few data points. Hence, we suggest to take the recall of this smell with care, and suggest that future studies should investigate this issue in more depth.

A further analysis of the false negatives shows that the smell detection missed findings because of imprecisions in the NLP libraries (i.e. Stanford NLP [71] for Lemmatization and POS Tagging and RFTagger [66] for morphologic analysis). For the dictionary-based smells, the lemmatization did not correctly deduce the correct lemma, e.g. it did not understand that a certain word was a plural of a lemma. If only the lemmatized version of the word, i.e. the singular form, is in the dictionary, then the smell detector does not correctly identify the smell. In the false negative cases for the **Comparative** and **Superlative Requirements Smell**, RFTagger did not correctly classify the inflection.

Interpretation. The study revealed that the precision strongly varies between the different smells. Qualitative analysis provided further insights described next.

We can now explain the high number of findings for vague pronouns in RQ 1. If we assume that a quarter of the findings are correct, the number of findings in this category is closer to the remaining smells. Also, we could see that while there are certain

reasons of imprecision that root from the study objects themselves and are, thus, unavoidable, there is plenty of space for optimization. First, existing techniques from NLP could be applied to improve certain smells, such as the **vague pronouns**. Second, from the examples we have seen, we would argue that the application of heuristics could heavily improve the precision of existing smell detection techniques. For example, if we exploit the information available from POS tagging, we can find out whether a comparison refers to a number or numerical expression.

Regarding recall, our analysis shows only a slight variance between the smells, with the only outlier being the **Superlative Requirements Smell**; however, since this is a very rare smell, this recall is based on only few data points, therefore, we must consider this result with care. When inspecting the reasons for false negatives, we found that optimizations could be made through the lemmatizer. Future research in this direction should compare whether the accuracy of lemmatizers as reported in the field of computational linguistics also holds for requirements engineering artifacts. Furthermore, we analyzed requirements in German language where lemmatization is a more difficult problem than in English, since the language makes stronger use of inflections (e.g. with cases or gender). Hence, smell detectors based on lemmatization for the English language might work better than the results indicate in our analysis.

In general, the precision and recall are therefore comparable to other approaches with related purposes (see Sect. 2). However, is it sufficient for an application of Requirements Smells in practice?

First, when looking at precision, we must take into account that the current state of practice consists still of manual work and that the cost for running an automatic analysis is virtually zero. Nevertheless, checking a false positive finding takes effort which an inspector could rather spend in reading the document in more detail. However, as we see a high variation in the precision over different smells, we need to discuss these separately. Several of the smells have a precision of 0.7 and higher which is considered acceptable in static code analysis [8]. For other Requirements Smells, the precision is below 0.5. This means that every other finding will be a false positive. This can

be critical in the effort spent in vain and annoy a user of the smell detection. Yet, we follow Menzies et al. [57] that a low precision can be still useful “When there is little or no cost in checking false alarms.” In our experience, the cost of checking a finding is often just a few seconds.

Second, when looking at recall, most of the smell detections reach a recall of more than 80%. Various publications, most prominently Kiyavitskaya [40] and Berry et al. [5], argue that a recall close to 100% is a basic requirement for any tool for automatic QA in RE. The core argument is that with a lower recall, reviewers stop checking these aspects and consequently miss defects, and that reviewers need to check the complete artifact anyway. However, if taking the example of spell checkers and grammar checks, these are still used on a daily basis, although they are far away from 100% recall. Therefore, one could consequently also argue that the precision is more important than the recall.

In any case, whether the reported precision and recall are sufficient in industry needs further research in the future. As mentioned above, it mainly depends on two factors: the required investment versus the gained benefit (similar to the concept of technical debt). For the required investment, we argue that, based on our experience of analyzing the various cases presented here, one can quickly iterate through the detected findings with low investment. To further support this discussion, the following research question analyzes the aspect of the benefits to practitioners in more detail.

Answer to RQ 2.1. As shown in Tables 7 and 8, and as shown in Fig. 10, the precision is on average around 59%, with an average recall of 82%, but both vary between smells. We consider this reasonable for a task that is usually performed manually. However, this also depends on the relevance of findings to practitioners, which we analyze in RQ 2.2. The study also reveals improvements for future work through the application of deeper NLP.

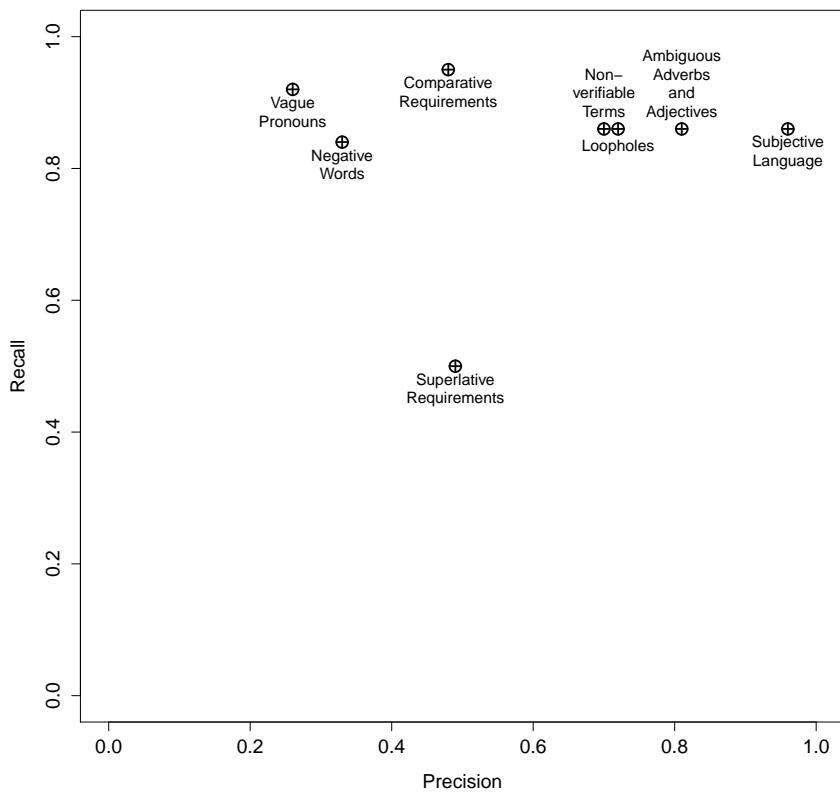


Figure 10: Precision and recall of the discussed smell detection approaches.

Table 8: Recall of smell detection within sample of 4 artifacts (16,271 words)

Smell	Findings in artifacts	Findings identified correctly	Recall
Ambiguity-related S.	74	64	0.86
Superlative Requirements S.	4	2	0.50
Comparative Requirements S.	21	20	0.95
Negative Words S.	64	54	0.84
Vague Pronouns S.	37	34	0.92
Average	40	34.8	0.82
Overall	200	174	0.87

6.2.4. RQ 2.2: Which of these smells are practically relevant in which context?

To understand whether the Requirements Smells help detecting relevant problems, we first performed a pre-study, in which we confronted practitioners of Daimler and Wacker with findings. The pre-study, which we reported in Femmer et al. [24], aimed at receiving qualitative and tacit feedback. It showed that Requirements Smells can in fact indicate relevant defects.

In contrast, in this study we analyze relevance in specific categories by interviewing practitioners at TechDivision on their opinion on the findings in terms of relevance, awareness, and whether these practitioners would resolve the suggested finding.

Quantitative observations. Table 9 reports the 20 findings that we discussed with TechDivision. In summary, we can see that they considered 65% of the findings as relevant for their context. Furthermore, they have not been aware of 45% of the findings. Lastly, they would act on 50% of the presented findings and on 40% even immediately.

Qualitative observations (true positives). The findings that the tool produces mostly constituted forms of underspecification. For example, in Finding #1 (see Table 9): "*As a searcher, I want to see the checkboxes in the different categories displayed more clearly, so that...* " (for similar examples, see Findings 3, 4, 14, 16, and 20). In this case, as in many of the other examples, the practitioners stated that no developer could implement this story properly. They also recalled various discussions in estimation meetings on what was to be done to complete these types of stories¹³.

In the previous research questions, we have seen that Requirements Smells are able to detect **loopholes** in requirements, such as the usage of the word *should*. To understand the relevance of this finding in the context of an agile company, we also discussed the **loophole** in Finding #6. When we pointed out the finding, they responded that they considered expressing what the system *should* do in user stories problematic. They considered this defect a low risk, as the developers understood ("*If you are told that you should take out the trash, you understand that it is an imperative.*") and their user stories did never turn out to be of legal relevance. They concluded that they want to avoid this, but it has no immediate urgency in a project situation.

ISO 29148 discusses the use of **negative statements** ("*capabilities not to be provided*"). In a previous study [24] practitioners expressed their reluctance of this criterion. In contrast, in this study, practitioners said they would act upon 2 out of 3 of the negative statements (Findings #9–11) that we presented to them as they revealed unclear requirements. In one case they even remembered that this led to discussions about the implementation during the sprint. Table 9 shows many more, similar examples.

Qualitative observations (false positives). Also interesting are those cases that practitioners considered not

¹³Note that discussions can have different objectives, i.e. *what* is to be implemented and *how*. For these, *how* to implement a story is the team's task and thus discussions can help finding the best way. In contrast, *what* the product owner wants is outside of the team's scope and therefore should not be a matter of discussion.

relevant in their context or where practitioners said they would not act upon. Summarized, the reasons were the following:

Domain and context knowledge: Some stories that were unclear to outsiders were understandable for someone knowing the system under consideration. For example, in user story #18 it was unclear to the first and second author what *their* refers to. It was clear, however, to both practitioners with knowledge about the system.

Process requirement: In Finding #8, the smell reveals another conspicuous finding: The developer should put as *low effort as possible* into the implementation of this story. In the discussion, the reason for this was that the customer did not want to pay much for this implementation. Thus the story should only be fulfilled if it was possible to be fulfilled cheaply. While the practitioners told us they would not change anything about this story, they agreed that the smell pointed out something that violates common user story practice.

Finding in reason part: In four cases, the practitioners agreed to the finding but considered it irrelevant as the finding was inside the *reason* part of the user story. This is due to this part of the user story only serving as additional information. This reason part is not used in testing nor is the information directly relevant for implementation. The main purpose is to understand the business value and to indicate the major goal to the team, similar to goals and goal modeling in traditional requirements engineering [50].

Answer to RQ 2.2. In summary, the practitioners expressed that 65% of the discussed findings were relevant, as they lead to lengthy discussions and unnecessary iterations in estimation. They also saw the problem of legal binding, but in contrast to the practitioners of Case A and B, they considered these findings less relevant. Due to these results, they expressed their strong interest in exploring smell detection for projects; we will explain the results of this discussion in RQ 4.

Further observations of quality defects in different parts of a user story

We considered especially the last explanation for rejecting findings (finding in reason part of a user story) particularly interesting. We had noticed that the reason part was often written in a rather imprecise way. To be able to quantify this aspect, we automatically split user stories according to the language patterns and quantified the distribution of words as well as findings over the different parts of user stories.

Table 10 shows the results of this analysis. The number of words is roughly distributed as follows: 11% of the words of a user story describe the role, 55% of the words describe the feature and 34% describe the reason. Of the 1,082 user stories, 290 had no reason part at all. Due to this uneven distribution, similar as in the previous analyses, we normalize the number of findings by the number of words in each part resulting in the *number of findings per 1,000 words*.

Only 1% of the findings are located in the role part. In fact, when we inspected these findings, they were false positives due to the grammatical problems described in the previous section. The absence of findings in this section is expected, as this part of the user story only names the role and does not offer many chances for smells as described in Sect. 3.2. For the remainder, 46% of the findings are located in the feature and 53% are located in the reason part. In relation to its size, the difference is striking: With 64 findings per 1,000 words, the reason has nearly double the number of findings of the feature part and nearly 70% more findings than the average requirement, as analyzed in Sect. 6.2.2.

In summary, the reason part of user stories is particularly prone to smells, but the qualitative analysis in RQ 2.2 reveals that practitioners consider findings in this section to be less relevant. This investigation could support further application of Requirements Smells in practice by helping to prioritize smells according to their location.

6.2.5. RQ 3: Which requirements quality defects can be detected with smells?

For 44 of the 51 requirements artifacts the students provided technical reviews. We qualitatively analyzed the results of 10 randomly selected reviews

(around 20%). The inspected reviews were conducted by 5–7 reviewers (mean: 5.6), took 90 minutes and resulted in 18–69 defects (mean: 38.1). We iterated through the 381 defects documented in the reviews and evaluated whether the smell detection produced findings indicating these defects. If no smell indicated the defect, we openly classified the defects. We did not quantify these results, because the resulting numbers would assume and suggest that the distribution of defects is representative for regular projects, which we are unsure about (i.e. because of a high number of spelling and grammatical issues).

The classification of the defects and their comparison with the detected smells resulted in the following list of defects indicated by Requirements Smells:

Sentence not understandable. In some instances, when the defect suggested changing the sentence to improve understandability, these sentences were highlighted especially by the **vague pronouns** and **negative statements** smells.

Improper legal binding. Various requirements artifacts had issues with improper legal binding. In one case, the reviewers recognized this and demanded the use of the term *must*. The **loop-holes** smell pinpointed at this issue.

Unspecified/unmeasurable NFRs. Various smells, especially the **superlatives smell**, indicated at defects of underspecification within non-functional requirements.

The remaining defects were not indicated by Requirements Smells.

Interpretation. The quantitative distribution of defects is not necessarily representative for industry projects and, thus, has not been analyzed. The reviews clearly show that manual inspection discovered the same defects as in the previous research question: Understandability, legally binding terminology and underspecified requirements. These are issues with regards to representation but also the content described in the artifact. We argue that these issues are common for requirements artifacts. Requirements Smells can therefore indicate relevant defects from multiple,

independent sources (manual inspection, interviews with practitioners, independent manual reviews) for multiple, independent cases.

Answer to RQ 3. Automatic smell detection can point to issues in both representation (e.g. improper legal binding) and content (underspecified/unmeasurable NFRs). The analysis of the reported defects indicates that more defects could be automatically detected (see section *further discussion on detectability of defects* described next). Nevertheless, just as for static code analysis, we see that automatic analysis can not indicate all defects and thus must be accompanied by reviews [73]. The fourth research question aims at analyzing this aspect in depth.

Further discussion on detectability of defects. During the analysis, if no smells indicated the defect, we openly classified the defects. While discussing the resulting list of defects and the degree to which they are detectable within the group of authors, we came up with a classification which is broader as initially planned while designing the study. This classification considers whether a defect:

- *Already can* be detected
- *Could* be detected, but is not implemented yet in our detection
- *Cannot be detected at the moment*, but should be soon
- *Cannot be detected at all* and probably won't be soon

This classification is purely based on our knowledge of existing related work and our subjective expectations gained during the data analysis process. The classification yielded in a map visualised in Fig. 11. The figure is structured in two dimensions: On the vertical axis, we group the defects into *defects relating to the content*, and *defects relating to representation*. Furthermore, on the horizontal axis, we map the items according to the expected precision and completeness we believe the detection could be (i.e. the classification above). The further left an item, the more precise and complete we expect a smell detection to be; the items on the right we assume to be close to impossible to detect in a general case.

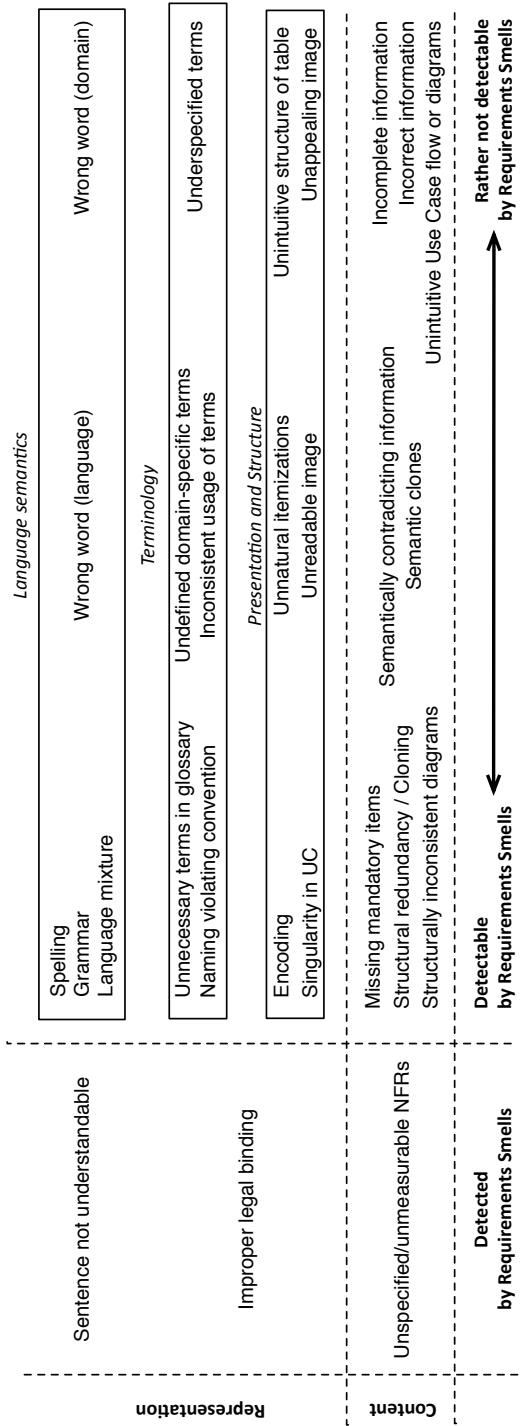


Figure 11: Findings in requirements reviews, classified by content/representation and detection

With the defects that our current approach does not reveal, this research question shows that more defects could be detected: These are namely defects with terminology, singularity in use cases and structural issues focusing on the content such as the absence of mandatory elements in the artifact [37], structural redundancy [34] or structural inconsistency between content. It remains unclear how far more enhanced language analysis with more sophisticated NLP and ontologies can enable to understand language. In any case, when a defect remains subtle and vague in its definition, such as an unintuitive structuring or design, we only see potential for automation if a defect can be defined precisely. For problems relating to the domain itself (e.g. incomplete information about the domain or incorrect information with regards to the domain), we consider it impossible to detect issues unless formalizing the concepts of the domain.

6.2.6. RQ 4: How could smells help in the QA process?

After the interviews and analysis, we asked all involved practitioners whether or not they think requirements smell detection is a helpful support, and whether and how they would integrate it in their context. We asked those questions openly and transcribed the answers for validation by the interviewees and later coding. In the following, we report on the results structured by topics. Where applicable, we provide the verbatim answers in relation to their cases (A, B or C).

Overall Evaluation. In general, all practitioners agreed on the usefulness of the smell detection even if considering different perspectives that arise from their process setting. One practitioner (Case C) reports that he expects one benefit in using smell detection is that it would lead to a reduction of the time spent for effort estimations (in context of agile methods), as the product owner could benefit from the smell detection on the fly and, thus, avoid misinterpretations later.

Quotes on Overall Evaluation	
	<i>"I think that smells can help to analyze a specification."</i>
B.	<i>"The method of Requirements Smells is a valuable extension in the area of requirements engineering and gives helpful input concerning the quality of specified requirements in early development phases."</i>
C.	<i>"I think such a smell detection is of high value to make sure that our team is confronted with already quality assured [user] stories. This can reduce the time in our effort estimations, because the product owner would directly notice on the fly what could lead to misinterpretations later."</i>

Integration into Process. When asked for how the practitioners would integrate the smell detection into their process setting, we got varying answers depending on the process. The practitioner relying more on rich process models (Case B) could imagine using a smell detection either as a support for the person writing the requirements or as part of a more fundamental QA method for the company. But also the practitioner relying more on the agile methods (Case C) could imagine using Requirements Smells as a support for the person writing the requirements or in context of analytical QA. In addition, one potential use is seen in context of problem management. Importantly, all practitioners see the full potential of a smell detection only if integrated in their existing tool chain (see also quotes on constraints and limitations).

Quotes on Integration into Process	
	<i>"I like to compare Requirements Smells to the "check spelling aid" known e.g. from Microsoft Word. So for me Requirements Smells are intuitive and lightweight and should be used and integrated within requirements engineering and quality assurance processes."</i>
C.	<i>"As a product owner, I would use a smell detection on the fly [...]. In addition, smell detection could help in analytical QA, as it could reveal when a problem occurs repeatedly, either in a project or in the company as a whole."</i>

Constraints and Limitations. One facet we consider especially interesting when using qualitative data is the chance to reveal further fields of improvement. We therefore concentrate now on the constraints that would hamper the usage of a smell detection. One facet we believe to be important is that practitioners want to avoid additional effort when using smell detection in their context. Furthermore, the practitioner of Case A believes that the automatic smell detection requires a common understanding on the notion of RE quality. He further indicates that the smell detection should explicitly take into account that some criteria cannot be met at every stage of a project.

Quotes on Constraints and Limitations

- "First, the people who need to write the specification received training which gives the required performance criteria. Second, abstraction levels must be taken into account during the smell detection process, since at higher abstraction levels different criteria cannot be met (e.g. vague pronouns or subjective language)."*
- B. *"As a product owner, I would use a smell detection on the fly provided that it would not mean additional effort [such as by having to use another tool]."*

Answer to RQ 4. Our practitioners provided a general agreement on potential benefits of using smell detection a quality assurance context. When asked how they would integrate the requirements smell detection, they see possibility for both analytical and constructive QA, provided, however, this integration would not increase the required effort, e.g. by integrating the detection into existing tool chains.

6.2.7. Evaluation of validity

We use the structure of threats to validity from [64] to discuss the evaluation of the validity of our study.

Construct validity. In our evaluation, we analyzed Requirements Smells in the terms of false positives, relevance and relation to quality defects. There are threats that the understanding of these terms varies and, thus, the results are not repeatable. Yet, we are

confident that our validity procedures described in Sect. 6.1.5 reduced this threat. For the false positives, we classified a subset of the findings independently, and afterwards compared (inter-rater agreement Cohen's kappa: 0.53) and discussed the results. We subsequently reclassified a different subset of findings again, which lead to an inter-rater agreement (Cohen's kappa) of 0.72. For the classification of false negatives, we reclassified one document separately, calculating the percentage of agreement on false positives¹⁴. This lead to an agreement of 88%.

We consider both of these substantial agreements, especially in the inherently ambiguous and complex domain of RE. Thus, we consider this threat as sufficiently controlled.

Internal validity. A threat to the internal validity of our results is that the experience of the students as well as the practitioners might play a role in their ratings of relevance or detection of quality defects. We mitigated this threat by choosing only practitioners for the ratings and interviews who had several years of experience. The students are only in the second year. We cannot mitigate this threat but consider the effect to be small. There might be some defects not found by the students that could have been indicated by a smell as well as unfound defects undetectable by smells. Hence, future studies will add to the classification but are unlikely to change it substantially. Personal pride could potentially have an impact on the answers to a RQ 2.2, if practitioners are not able to professionally discuss their own work products. In our cases, however, all practitioners openly accepted the discussions (as can be seen in their answers). Even though we carefully supervised this threat, we have not found signs of personal bias in the cases involved. Finally, the students might also have been influenced by the review guidelines we provided. Yet, none of the investigated smells was explicitly listed in the

¹⁴We did not employ Cohen's kappa here, since the number of true positives (non-smell words) would strongly dominate the result and therefore skew the inter-rater agreement. Instead, we calculated the ratio of findings which both rating teams independently classified as false positive to the number of findings which only one of the teams classified false positive.

guidelines. Instead, the guideline contained rather high-level aspects such as “unambiguity”. Although we consider this threat to be a minor one, it is still present.

External validity. As requirements engineering is a diverse field, the main threat to the external validity of our results is that we do not cover all domains and ways of specifying requirements. We mitigated this threat to some degree by covering at least several different domains and study objects, of which some are purely textual requirements artifacts, some use cases, and some user stories. We argue that this represents a large share of today’s requirements practices.

Reliability. Our study contains several classifications and ratings performed by people. This constitutes a threat to the reliability of our results. We are confident, however, that the peer debriefing and member checking procedures helped to reduce this threat.

7. Conclusion

In this paper, we defined Requirements Smells and presented an approach to the detection of Requirements Smells which we empirically evaluated in a multi-case study. In the following, we summarize our conclusions, relate it to existing evidence on the detection of natural language quality defects in requirements artifacts, and we discuss the impact and limitations of our approach and its evaluation. We close with outlining future work.

7.1. Summary of conclusions

First, we proposed a light-weight approach to detect Requirements Smells. It is based on the natural language criteria of ISO 29148 and serves to rapidly detect Requirements Smells. We define the term *Requirement Smell* as an indicator of a quality violation, which may lead to a defect, with a concrete location and a detection mechanism, and we also give definitions of a concrete set of smells.

Second, we developed an implementation that is able to detect Requirements Smells by using part-of-speech (POS) tagging, morphological analysis and dictionaries. We found that it is possible to provide

such tool support and outlined how such a tool could be integrated into quality assurance.

Third, in the empirical evaluation, our approach showed to support us in automatically analysing requirements of the size of 250k words. Findings were present throughout all cases but in varying frequencies between 22 and 67 findings per 1,000 words. Outliers indicated serious issues. An investigation of the detection precision showed an average precision around 0.59 over all smells, again varying between 0.26 and 0.96. The recall was on average 0.82, but also varied between 0.5 and 0.95. To improve the accuracy, we described concrete improvement potential based on real world, practical examples.

A further analysis of reviews and practitioner’s opinions strengthen our confidence that smells indicate quality defects in requirements. For these quality defects, practitioners explicitly stated the negative impact of discovered findings on estimation and implementation in projects. The study also showed, however, that while Requirements Smell detection can help during QA presumably in a broad spectrum of methodologies followed (including agile ones), the relevance of Requirements Smells varies between cases. Hence, it is necessary to tailor the detection to the context of a project or company. We analyzed this factor in depth, demonstrating that the reason part of a user story contains most findings (absolutely and relatively), but practitioners consider these findings less relevant as they argue that this part is not commonly used in implementation or testing. This raises the question of the relevance of this part at all, at least from a quality assurance perspective, which should be investigated in future work.

Our comparison with defects found in reviews furthermore showed that the Requirements Smell detection partly overlaps with results from reviews. As a result, we provide a map of defects in requirements artifacts in which we give a first indication where Requirements Smells can provide support and where they cannot.

Therefore, we provide empirical evidence from multiple, independent sources (manual inspection, interviews with practitioners, independent manual reviews) for multiple, independent cases, showing that Requirements Smells can indicate relevant defects across dif-

ferent forms of requirements, different domains, and different methodologies followed.

7.2. Relation to existing evidence

Existing approaches in the direction of automatic QA for RE are based on various quality models, including the ambiguity handbook by Berry et al. [7], the now superseeded IEEE 830 standard [32] and proprietary models. Yet, according to a recent literature review by Schneider and Berenbach [67], ISO 29148 is the current standard in RE “*that every requirements engineer should be familiar with*”. However, no detailed empirical studies (see Table 1) exist for the quality violations described in ISO 29148. When comparing to similar, related quality violations, also few empirical, industrial case studies exist (see Table 2). Gleich et al. [30] and Chantree et al. [11] report for conceptually similar problems, a precision of the detection between 34% and 75% (97% in a special case), and a recall between 2% and 86%. Krisch and Houdek [49] report a lower precision in an industrial setting. The precision and recall for the detection of the smells, which we developed based on the description in the standard, are in a similar range to the aforementioned. In summary, this work provides a detailed empirical evaluation on the quality factors of ISO 29148, including a deeper understanding of both existing and novel factors.

We also take a first step from the opposite perspective: So far, to all our knowledge, all related work starts from a certain quality model and goes into automation. Our results to RQ 3 provides a bigger picture for understanding in how far quality defects in requirements could be addressed through automatic analysis in general.

Our results to RQ 2.2 furthermore provides evidence for the claim by Gervasi and Nuseibeh [29] that “*Lightweight validation can discover subtle errors in requirements.*” More precisely, our work indicates that automatic analysis can find a set of relevant defects in requirements artifacts by providing evidence from multiple case studies in various domains and approaches. The responses by practitioners to the findings do, to some extent, contradict the claim by Kiyavitskaya et al. [40] who state that “*any tool [...] should have 100% recall*”. Practitioners responded

very positively on our first prototype and the smells it finds. Yet, obviously, more detailed and broader evaluations, especially conducted independently by other researchers not involved in the development of Smella, should follow.

7.3. Impact/Implications

For practitioners, Requirements Smells provide a way to find certain issues in a requirements artifact without expensive review cycles. We see three main benefits of this approach: First, the approach, just as static analysis for code, can enable project leads to keep a basic hygiene for their requirements artifacts. Second, the review team can avoid discussing obvious issues and focus on the important, difficult, domain-specific aspects in the review itself. Third, the requirements engineers receive a tool for immediate feedback, which can help them to increase their awareness for certain quality aspects and establish common guidelines for requirements artifacts.

Yet, the low precision for some of the smells might cause unnecessary work checking and rejecting findings from the automatic smell detection. Hence, at least for now, it is advisable to concentrate on the highly accurate smells.

For researchers, this work sharpens the term Requirements Smell by providing a definition and a taxonomy. By implementing and rating concrete smell findings, we also came to the conclusion, however, that not all of the requirements defects from ISO/IEC/-IEEE 29148 can be clearly distinguished as Requirements Smells. In particular, the difference between *Subjective Language*, *Ambiguous Adverbs and Adjectives*, *Non-verifiable Terms*, and *Loopholes* was not always clear to us during our investigations (see RQ 2.1). Therefore, we, as a community, can take our smell taxonomy as a starting point, but we also need to critically reflect on some smells to further refine the taxonomy.

Finally, empirical evidence in RE is, in general, difficult to obtain because many concepts depend on subjectivity [55]. One issue increasing the level of difficulty in evidence-based research in RE remains that most requirements specifications are written in natural language. Therefore, they do not lend themselves for automated analyses. Requirements Smell

detection provides us with a means to quantify the extent of certain defects in a large sample of requirements artifacts while explicitly taking into account the sensitivity of findings to their context. Hence, this allows us to consider a whole new spectrum of questions worth studying in an empirical manner.

7.4. Limitations

We concentrated on a first set of concrete Requirements Smells based on our interpretation of the sometimes imprecise language criteria of ISO/IEC/IEEE 29148. There are more smells, also with different characteristics than the ones we proposed and analyzed. In addition, even though we diversified our study objects over domains, methods and different types of requirements, we cannot generalize our findings to all applicable contexts. We therefore consider the presented results only a first step towards the continuous application of Requirements Smells in software engineering projects.

7.5. Future work

Our work focuses on Requirements Smells based on ISO/IEC/IEEE 29148. Future work needs to clarify and extend this taxonomy based on related work and experience in practice. This also includes the development of other Requirements Smell detection techniques to increase our understanding about which defects can be revealed by Requirements Smells and which defects cannot.

Second, this first study gained first insights into the usefulness of Requirements Smells for QA. We furthermore sketched an integration of Requirements Smells into a QA process. Yet, a full integration and the consequences must be analyzed in depth. In particular, we need to understand whether smell detection as a supporting tool, similar to spell checking, as pointed out by one of our participants, enables requirements engineers to improve their requirements artifacts.

Lastly, Requirements Smells focus on the detection of issues in requirements artifacts. They require a thorough understanding of the impact of a quality defect, which is hence also part of the requirements smell taxonomy. This link must be carefully evaluated and analyzed in practice. Our preliminary works on this topic [23, 59] provide first ideas in that direction.

Acknowledgments

We would like to thank Elmar Juergens, Michael Klose, Ilona Zimmer, Joerg Zimmer, Heike Frank, Jonas Eckhardt as well as the software engineering students of Stuttgart University for their support during the case studies and feedback on earlier drafts of this paper.

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

Bibliography

References

- [1] V. Ambriola and V. Gervasi. On the systematic analysis of natural language requirements with CIRCE. *Automated Software Engineering*, 13(1):107–167, 2006.
- [2] B. Anda and D. I. K. Sjøberg. Towards an inspection technique for use case models. In *Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering*. ACM, 2002.
- [3] D. J. Anderson. *Kanban*. Blue Hole Press, 2010.
- [4] C. Arora, M. Sabetzadeh, L. Briand, and F. Zimmer. Automated Checking of Conformance to Requirements Templates using Natural Language Processing. *IEEE Transactions on Software Engineering*, 41(10):944–968, 2015.
- [5] D. Berry, R. Gacitua, P. Sawyer, and S. F. Tjong. The case for dumb requirements engineering tools. In *Requirements Engineering: Foundation for Software Quality*, pages 211–217. Springer Berlin Heidelberg, 2012.
- [6] D. M. Berry, A. Bucciarone, S. Gnesi, G. Lami, and G. Trentanni. A new quality model for natural language requirements specifications. In *Requirements Engineering: Foundation for Software Quality*. Essener Informatik Beiträge, 2006.

- [7] D. M. Berry, E. Kamsties, and M. M. Krieger. From Contract Drafting to Software Specification : Linguistic Sources of Ambiguity. Technical report, School of Computer Science, University of Waterloo, Waterloo, ON, Canada, 2003.
- [8] A. Bessey, K. Block, B. Chelf, A. Chou, B. Fulton, S. Hallem, C. Henri-Gros, A. Kamsky, S. McPeak, and D. Engler. A few billion lines of code later: using static analysis to find bugs in the real world. *Communications of the ACM*, 53(2):66–75, 2010.
- [9] S. Bradner. Key words for use in RFCs to Indicate Requirement Levels - RFC 2119. <https://www.ietf.org/rfc/rfc2119.txt>, 1997.
- [10] A. Bucchiarone, S. Gnesi, and P. Pierini. Quality Analysis of NL Requirements : An Industrial Case Study. In *13th IEEE International Requirements Engineering Conference*, pages 390–394, 2005.
- [11] F. Chantree, B. Nuseibeh, A. D. Roeck, and A. Willis. Identifying Nocuous Ambiguities in Natural Language Requirements. *14th IEEE International Requirements Engineering Conference*, pages 59–68, sep 2006.
- [12] A. Ciemniewska, J. Jurkiewicz, L. Olek, and J. Nawrocki. Supporting Use-Case Reviews. In *Business Information Systems*, pages 424–437. Springer Berlin Heidelberg, 2007.
- [13] A. Cockburn. *Writing Effective Use Cases*. Addison-Wesley, 2000.
- [14] M. Cohn. *User stories applied: For agile software development*. Addison-Wesley Professional, 2004.
- [15] A. Davis, S. Overmyer, K. Jordan, J. Caruso, F. Dandashi, A. Dinh, G. Kincaid, G. Ledeboer, P. Reynolds, P. Sitaram, A. Ta, and M. Theofanous. Identifying and measuring quality in a software requirements specification. In *Proceedings First International Software Metrics Symposium*, pages 141–152, 1993.
- [16] F. De Bruijn and H. L. Dekkers. Ambiguity in natural language software requirements: A case study. In *Requirements Engineering: Foundation for Software Quality*, pages 233–247. Springer Berlin Heidelberg, 2010.
- [17] C. Denger, D. Berry, and E. Kamsties. Higher quality requirements specifications through natural language patterns. In *Software: Science, Technology and Engineering*, pages 80–90. IEEE, 2003.
- [18] F. Fabbrini, M. Fusani, S. Gnesi, and G. Lami. An automatic quality evaluation for natural language requirements. In *Proceedings of the Seventh International Workshop on Requirements Engineering: Foundation for Software Quality*, volume 1, pages 4–5, 2001.
- [19] F. Fabbrini, M. Fusani, S. Gnesi, and G. Lami. The linguistic approach to the natural language requirements quality: benefit of the use of an automatic tool. In *Proceedings 26th Annual NASA Goddard Software Engineering Workshop*, pages 97–105. IEEE Computer Society, 2001.
- [20] M. Fagan. Design and code inspections to reduce errors in program development. In *Software pioneers*, pages 575–607. Springer, 2002.
- [21] D. Falessi, G. Cantone, and G. Canfora. Empirical Principles and an Industrial Case Study in Retrieving Equivalent Requirements via Natural Language Processing Techniques. *IEEE Transactions on Software Engineering*, 39(1):18–44, 2013.
- [22] A. Fantechi, S. Gnesi, G. Lami, and A. Maccari. Application of linguistic techniques for Use Case analysis. *Requirements Engineering*, 8(3):161–170, 2003.
- [23] H. Femmer, J. Kučera, and A. Vetrò. 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*, ESEM ’14, pages 21:1–21:4, New York, NY, USA, 2014. ACM.

- [24] H. Femmer, D. Méndez Fernández, E. Juergens, M. Klose, I. Zimmer, and J. Zimmer. Rapid requirements checks with requirements smells: Two case studies. In *Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering*, RCoSE 2014, pages 10–19, New York, NY, USA, 2014. ACM.
- [25] H. Femmer, D. Méndez Fernández, S. Wagner, and S. Eder. Supplementary online material: Analysis of related work. Created on: 2015-12-22.
- [26] H. Femmer, J. Mund, and D. Mendez Fernandez. It's the Activities, Stupid! A New Perspective on RE Quality. In *Proceedings of the 2nd International Workshop on Requirements Engineering and Testing*, pages 13–19, 2015.
- [27] M. Fowler and K. Beck. *Refactoring: improving the design of existing code*. Addison-Wesley Professional, 1999.
- [28] G. Génova, J. M. Fuentes, J. Llorens, O. Hurtado, and V. Moreno. A framework to measure and improve the quality of textual requirements. *Requirements Engineering*, 18(1):25–41, Sept. 2011.
- [29] V. Gervasi and B. Nuseibeh. Lightweight validation of natural language requirements. *Software: Practice and Experience*, 32(2):113–133, Feb. 2002.
- [30] B. Gleich, O. Creighton, and L. Kof. Ambiguity detection: Towards a tool explaining ambiguity sources. In *Requirements Engineering: Foundation for Software Quality*, volume 6182, pages 218–232. Springer Berlin Heidelberg, 2010.
- [31] B. Hauptmann, M. Junker, S. Eder, L. Heinemann, R. Vaas, and P. Braun. Hunting for smells in natural language tests. In *Proceedings of the International Conference on Software Engineering*, pages 1217–1220, 2013.
- [32] IEEE Computer Society. IEEE Recommended Practice for Software Requirements Specifications. <https://standards.ieee.org/findstds/standard/830-1998.html>, 1998.
- [33] ISO, IEC, and IEEE. ISO/IEC/IEEE 29148:2011. <https://standards.ieee.org/findstds/standard/29148-2011.html>, 2011.
- [34] E. Juergens, F. Deissenboeck, M. Feilkas, B. Hummel, B. Schaetz, S. Wagner, C. Domann, and J. Streit. Can Clone Detection Support Quality Assessments of Requirements Specifications? In *Proceedings of the International Conference on Software Engineering*, pages 79–88, 2010.
- [35] E. Juergens, F. Deissenboeck, B. Hummel, and S. Wagner. Do code clones matter? In *Proceedings of the International Conference on Software Engineering*, pages 485–495, 2009.
- [36] D. Jurafsky and J. H. Martin. *Speech and Language Processing*. Pearson Education, 2nd edition, 2014.
- [37] M. I. Kamata and T. Tamai. How Does Requirements Quality Relate to Project Success or Failure? In *15th IEEE International Requirements Engineering Conference*, pages 69–78, 2007.
- [38] E. Kamsties, D. M. Berry, and B. Paech. Detecting Ambiguities in Requirements Documents Using Inspections. In *Proceedings of the 1st Workshop on Inspection in Software Engineering*, pages 68–80, 2001.
- [39] E. Kamsties and B. Peach. Taming ambiguity in natural language requirements. In *Proceedings of the International Conference on System and Software Engineering and their Applications*, pages 1–8, 2000.
- [40] N. Kiyavitskaya, N. Zeni, L. Mich, and D. M. Berry. Requirements for tools for ambiguity identification and measurement in natural language requirements specifications. *Requirements Engineering*, 13(3):207–239, 2008.
- [41] E. Knauss and T. Flohr. Managing requirement engineering processes by adapted quality gateways and critique-based RE-tools. In *Proceedings of Workshop on Measuring Requirements for Project and Product Success*, Nov. 2007.

- [42] E. Knauss, D. Lübke, and S. Meyer. Feedback-Driven Requirements Engineering : The Heuristic Requirements Assistant. In *Proceedings of the International Conference in Software Engineering*, pages 587–590, 2009.
- [43] J. C. Knight and E. A. Myers. An improved inspection technique. *Communications of the ACM*, 36(11):51–61, 1993.
- [44] L. Kof. Scenarios: Identifying missing objects and actions by means of computational linguistics. In *In Proceedings of the 15th IEEE International Requirements Engineering Conference*, pages 121–130, Oct 2007.
- [45] L. Kof. Treatment of Passive Voice and Conjunctions in Use Case Documents. *Natural Language Processing and Information Systems*, 4592:181–192, 2007.
- [46] S. J. Körner and T. Brumm. Improving natural language specifications with ontologies. In *Proceedings of the 21st International Conference on Software Engineering and Knowledge Engineering*, pages 552–557. World Scientific, 2009.
- [47] S. J. Körner and T. Brumm. Natural Language Specification Improvement With Ontologies. *International Journal of Semantic Computing*, 03(04):445–470, 2009.
- [48] S. J. Körner and T. Brumm. RESI - A natural language specification improver. In *Proceedings of the 2009 IEEE International Conference on Semantic Computing*, pages 1–8. IEEE, 2009.
- [49] J. Krisch and F. Houdek. The Myth of Bad Passive Voice and Weak Words: An Empirical Investigation in the Automotive Industry. In *23rd IEEE International Requirements Engineering Conference*, pages 344–351, 2015.
- [50] A. V. Lamsweerde. *Requirements Engineering*. John Wiley & Sons, 2009.
- [51] G. Lucassen, F. Dalpiaz, S. Brinkkemper, and J. van der Werf. Forging High-Quality User Stories: Towards a Discipline for Agile Requirements. In *23rd IEEE International Requirements Engineering Conference*, pages 126–135, 2015.
- [52] A. D. Lucia, F. Fasano, R. Oliveto, and G. Tortora. Recovering traceability links in software artifact management systems using information retrieval methods. *ACM Transactions on Software Engineering and Methodology*, 16(4), Sept. 2007.
- [53] J. Ludewig and H. Licher. *Software Engineering*. dpunkt.verlag, 2nd edition, 2010.
- [54] A. Mavin, P. Wilkinson, A. Harwood, and M. Novak. EARS (Easy Approach to Requirements Syntax). *Proceedings of the IEEE International Conference on Requirements Engineering*, pages 317–322, 2009.
- [55] D. Méndez Fernández, J. Mund, H. Femmer, and A. Vetrò. In Quest for Requirements Engineering Oracles: Dependent Variables and Measurements for (good) RE. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, pages 3:1–3:10. ACM, 2014.
- [56] D. Méndez Fernández and S. Wagner. Naming the Pain in Requirements Engineering: A Design for a Global Family of Surveys and First Results from Germany. *Information and Software Technology*, 57(1):616–643, 2015.
- [57] T. Menzies, A. Dekhtyar, J. Distefano, and J. Greenwald. Problems with precision: A response to "Comments on 'data mining static code attributes to learn defect predictors'". *IEEE Transactions on Software Engineering*, 33(9):637–640, 2007.
- [58] L. Mich, M. Franch, and P. L. Novi Inverardi. Market research for requirements analysis using linguistic tools. *Requirements Engineering*, 9(2):151–151, 2004.
- [59] J. Mund, H. Femmer, D. Méndez Fernández, and J. Eckhardt. Does Quality of Requirements Specifications matter? Combined Results of Two Empirical Studies. In *Proc. of the 9th International*

- Symposium on Empirical Software Engineering and Measurement*, pages 1–10, 2015.
- [60] D. Parachuri, A. Sajeev, and R. Shukla. An Empirical Study of Structural Defects in Industrial Use-cases. In *Proceedings of the International Conference on Software Engineering*, pages 14–23. ACM, 2014.
 - [61] M. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.
 - [62] A. Rago, C. Marcos, and J. A. Diaz-Pace. Identifying duplicate functionality in textual use cases by aligning semantic actions. *Software & Systems Modeling*, pages 1–25, Aug. 2014.
 - [63] P. Runeson and M. Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14(2):131–164, Dec. 2008.
 - [64] P. Runeson, M. Höst, A. Rainer, and B. Regnell. *Case Study Research in Software Engineering. Guidelines and Examples*. Wiley, 2012.
 - [65] F. Salger. Requirements reviews revisited: Residual challenges and open research questions. In *Proceedings of the 2013 21st IEEE International Requirements Engineering Conference*, pages 250–255. IEEE, 2013.
 - [66] H. Schmid and F. Laws. Estimation of conditional probabilities with decision trees and an application to fine-grained POS tagging. In *Proceedings of the Conference on Computational Linguistics*, pages 777–784. Association for Computational Linguistics, 2008.
 - [67] F. Schneider and B. Berenbach. A Literature Survey on International Standards for Systems Requirements Engineering. In *Proceedings of the Conference on Systems Engineering Research*, volume 16, pages 796–805, Jan. 2013.
 - [68] K. Schwaber and J. Sutherland. The scrum guide. Technical report, Scrum.org, 2011.
 - [69] F. Shull, I. Rus, and V. Basili. How perspective-based reading can improve requirements inspections. *Computer*, 33(7):73–79, 2000.
 - [70] S. F. Tjong and D. M. Berry. The design of SREE - A prototype potential ambiguity finder for requirements specifications and lessons learned. In *REFSQ*, pages 80–95. Springer Berlin Heidelberg, 2013.
 - [71] K. Toutanova, D. Klein, and C. D. Manning. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, 1(June):252–259, 2003.
 - [72] A. van Deursen, L. Moonen, A. van den Bergh, and G. Kok. *Refactoring test code*. CWI, 2001.
 - [73] S. Wagner, J. Jürjens, C. Koller, and P. Trischberger. Comparing bug finding tools with reviews and tests. In *Proceedings of Testing of Communicating Systems*, pages 40–55. Springer, 2005.
 - [74] W. M. Wilson, L. H. Rosenberg, and L. E. Hyatt. Automated analysis of requirement specifications. In *Proceedings of the International Conference on Software Engineering*, pages 161–171. ACM, 1997.
 - [75] M. V. Zelkowitz, R. Yeh, R. G. Hamlet, J. D. Gannon, and V. R. Basili. The Software Industry: A State of the Art Survey. *Foundations of Empirical Software Engineering: The Legacy of Victor R. Basili*, 1:383–383, 1983.
 - [76] M. Zhang, T. Hall, and N. Baddoo. Code bad smells: a review of current knowledge. *Journal of Software Maintenance and Evolution*, 23(3):179–202, 2011.

Appendix A. Requirements Checklist

Table 4: Study objects

Artifact	Topic	Size in Words	# Requirements	# Use Cases	# User Stories
A1	Adaptive valve control	1896	91		
A2	Exhaust control	2244	72		
A3	Driving information	199	12		
A4	Engine startup control	975	44		
A5	Engine control	524	49		
A6	Powertrain communication	1100	55		
Sum Daimler		6938	323		
B1	Management of access control	2093	9	18	
B2	Event notification	1015	3	19	
B3	Document management	458	1	16	
Sum Wacker		3566	13	53	
C1	Webshop for fashion articles	5226		168	
C2	CMS in transportation domain	2742		123	
C3	CRM system	6863		230	
C4	Webshop for hardware articles	13124		561	
Sum TechDivision		27955		1082	
Avg Stuttgart		4470		18.9	
Sum Stuttgart		227973		966	
Sum over all		266432	336	53	1082

Table 6: Quantitative summary of smell findings

Case	Num Words	All Smells	Subjective Language Smell	Loophole Smell	Vague Pronouns Smell	Superlatives Smell	Negative Words Smell	Comparatives Smell	Non-verifiable Smell	Ambiguous A & Smell
	abs	rel	abs	rel	abs	rel	abs	rel	abs	abs
A1	1896	45	23.7	4	2.11	2	1.05	13	6.86	7
A2	2244	52	23.2	6	2.67	3	1.34	20	8.91	1
A3	199	5	25.1	0	0	0	3	15.08	0	0
A4	975	29	29.7	3	3.08	1	1.03	15	15.38	0
A5	524	20	38.2	0	0	0	0	26.72	0	0
A6	1100	32	29.1	0	0	0	8	7.27	0	0
Sum Daimler	6938	183	26.4	13	1.87	6	0.86	73	10.52	8
									1.15	53
									7.64	20
									2.88	8
									1.15	2
									0.29	
B1	2093	90	43	5	2.39	11	5.26	40	19.11	6
B2	1015	28	27.6	2	1.97	1	0.99	13	12.81	0
B3	458	31	67.7	0	0	19	41.48	9	19.65	1
Sum Wacker	3566	149	41.8	7	1.96	31	8.69	62	17.39	7
									1.96	23
									6.45	17
									4.77	1
									0.28	1
C1	5226	229	43.8	48	9.18	5	0.96	104	19	3
C2	2742	120	43.8	11	4.01	7	2.55	62	22.61	3
C3	6863	233	34	30	4.37	14	2.04	105	15	0.09
C4	13124	572	43.6	35	2.67	16	1.22	339	25.83	11
Sum TechDivision	27955	1154	41.3	124	4.44	42	1	610	21.82	23
									0.57	0.82
									174	154
									5.51	11
									0.39	16
									0.57	
Mean Stuttgart	4470	198.5	44.4	6.45	1.44	19.65	4	117.37	26.26	5.12
Sum Stuttgart	227973	10122	44.4	329	1.44	1002	4	5986	26.26	261
Over all	266432	11608	43.6	473	1.78	1081	4.06	6731	25.26	299

Table 7: Precision of smell detection

Smell	Findings Inspected	Findings Accepted	Findings Rejected	Precision
Subjective Language Smell	69	66	3	0.96
Ambiguous Adverbs and Adjectives Smell	21	17	4	0.81
Loophole Smell	60	43	17	0.72
Non-verifiable Term Smell	23	16	7	0.70
Superlative Requirements Smell	39	19	20	0.49
Comparative Requirements Smell	88	42	46	0.48
Negative Words Smell	129	42	87	0.33
Vague Pronouns Smell	187	48	139	0.26
Average	77	36.6	40.4	0.59
Overall	616	293	323	0.48

Table 9: Exemplary findings; shortened and translated by the authors, findings in bold

ID	Finding	Relevant?	Aware?	Resolve?
1	As a visitor, I want to see the checkboxes in the different categories displayed more clearly , so that I can see more quickly that I can select and deselect categories.	Yes	Yes	Yes in short term
2	As a visitor, I want to see the checkboxes in the different categories displayed more clearly, so that I can see more quickly that I can select and deselect categories.	No	No	No
3	As an editor, I want to make it simpler to differentiate between ...	No	No	No
4	As a visitor, I want to see further details , e.g. (...), so that ...	Yes	Yes	Yes immediately
5	As a customer, I want, if I have a larger number of E-Mails in my mailbox, ...	Yes	Yes	Yes immediately
6	As an editor, I want to make it simpler to differentiate between A and B, therefore, A should be labeled as ...	Yes	No	Yes in long term
7	As a provider, I want that, as far as possible , all fields, are mapped between System A and System B.	Yes	Yes	Yes immediately
8	As a provider I want the news section to be implemented with an effort, as low as possible .	Yes	Yes	No
9	As a visitor, I do not want to see category X, so that I am not confronted with the issue.	No	No	No
10	As a visitor of the webpage, I want for not selected categories, the displayed hearts of the score (search results list) to be displayed in such a color, that the score display is not changed and always only hearts of relevant categories are displayed in color.	Yes	Yes	Yes immediately
11	As an employee, I want that an article, if no price is imported , despite the label 'available' to be not displayed in SYSTEM X, so that the article automatically resumes when a price is imported.	Yes	Yes	Yes immediately
12	As a user, I want to have the possibility to browse through previous and next products, so that I can quickly and easily look at multiple product without having to go back to the overview page.	No	No	No
13	As a visitor, I want to navigate to meaningfully structured categories via the menus.	Yes	Yes	Yes immediately
14	As a visitor, I want to open the pictures of the website, so that unnecessary waiting is avoided.	Yes	Yes	Yes immediately
15	As a visitor, I want to quickly designed search-suggest-box when I enter a text and wait.	Yes	Yes	Yes immediately
16	As a visitor of the website, I want a nice designed search-suggest-box when I enter a text and wait.	Yes	Yes	Yes immediately
17	As a buyer, I want to select from a set of shopping providers (...), so that I can select the best suited shopping provider.	No	No	No
18	As an editor, I want to have multiple entry points for linking categories, so that the visitor can (...) get an overview of selected brands and categories and their filters.	Yes	No	No
19	As [OTHER SYSTEM], I want that an order of the status 'Order income' transitions into status 'wait for transmission into [SYSTEM]', so that I do not see the order when indexing open orders and so I do not process the order multiple times and that one can see the status of the order in the backend properly).	No	No	No
20	As an editor, I want to know a good way how to transfer news content from [SYSTEM] to [SYSTEM] to be able to efficiently migrate everything at once.	Yes	Yes	No

Table 10: Findings in different parts of user stories (T=total, Ro=role, F=feature, Re=reason)

Case	#Stories	w/o reason	Size in Words			Findings absolute			Findings per 1,000 Words			
			Total	Role	Feature	Reason	T.	Ro.	F.	Re.	T.	Ro.
C1	168	23	5226	801	2375	2050	229	1	83	145	44	1
C2	123	45	2742	260	1552	930	120	0	62	58	44	0
C3	230	19	6863	824	3090	2949	233	5	81	147	34	6
C4	561	203	13124	1188	8223	3713	572	0	307	265	44	0
Sum	1082	290	27955	3073	15240	9642	1154	6	533	615	41	2
											35	64

Table A.11: Checklist for the students' requirements reviews. Created by Anke Drappa, Patricia Mandl-Striegnitz and Holger Röder based on [13] and [53]. Translated from German.

The document is well structured and easy to understand.
All used terms are clearly defined and consistently used.
All external interfaces are clearly defined.
The level of detail is consistent throughout the document.
The requirements are consistent and unambiguous.
The defined requirements are consistent with the state of the art.
All tasks and data have useful identifiers.
Data is not defined redundantly.
The defined relationships between data objects are necessary and sufficient.
The specification of quality attributes is realistic, useful, quantifiable and unambiguous.
The user interface is comfortable and easy to learn.
The use case describes a behaviour of the system which is valuable and visible for the actor.
The use case is described in a table which is consistently used for the whole requirements specification.
The use case has a unique ID.
The use case has a unique and expressive name.
The main actor's goal is described in an understandable way.
All actors participating in the use case are specified.
If there is more than one actor, the main actor is identified.
The preconditions of the use case are specified.
The postconditions for the use case are specified.
It is clearly specified how the main actor triggers the main success scenario.
The main success scenario has 3 to 9 steps.
After the main success scenario, the postconditions hold.
The main actor reaches their goal by the main success scenario.
Each step is sequentially numbered.
It is clear which actor is executing the step.
The step does not describe details of the user interface.
The step describes exactly one action of the acting actor.
There are postconditions for each extension.
It is clearly specified in which step the main success scenario deviates into an extension.
The conditions for the deviation into an extension are clearly specified.
After an extension, all postconditions for that extension hold.