Rapid Requirements Checks with Requirements Smells: Two Case Studies

Conference Paper · June 2014

DOI: 10.1145/2593812.2593817

Rapid Requirements Checks with Requirements Smells: Two Case Studies

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

Daniel Méndez Fernández
Technische Universität München, Germany
mendezfe@in.tum.de

Elmar Juergens
CQSE GmbH, Germany
juergens@cqse.eu

Michael Klose
Wacker Chemie AG, Germany
michael.klose@wacker.com

Ilona Zimmer
MBtech Group GmbH & Co. KGaA, Germany
ilona.zimmer@mbtech-group.com

Jörg Zimmer
Daimler AG, Germany
joerg.zimmer@daimler.com

ABSTRACT

Bad requirements quality can have expensive consequences during the software development lifecycle. Especially, if iterations are long and feedback comes late – the faster a problem is found, the cheaper it is to fix.

We propose to detect issues in requirements based on requirements (bad) smells by applying a light-weight static requirements analysis. This light-weight technique allows for instant checks as soon as a requirement is written down. In this paper, we derive a set of smells, including automatic smell detection, from the natural language criteria of the ISO/IEC/IEEE 29148 standard.

We evaluated the approach with 336 requirements and 53 use cases from 9 specifications that were written by the car manufacturer Daimler AG and the chemical business company Wacker Chemie AG, and discussed the results with their requirements and domain experts.

While not all problems can be detected, the case study shows that lightweight smell analysis can uncover many practically relevant requirements defects. Based on these results and the discussion with our industry partners, we conclude that requirements smells can serve as an efficient supplement to traditional reviews or team discussions, in order to create fast feedback on requirements quality.

Categories and Subject Descriptors

D.2.1 [**Software Engineering**]: Requirements/Specification

General Terms

Requirements Engineering, Quality Assurance, Natural Language

ICSE '14 Hyderabad, India

Keywords

Requirements Engineering, Analytical Quality Assurance, Requirements Smells

1. INTRODUCTION

Issues in requirements, such as ambiguities or incomplete requirements specifications, can lead to time and cost overrun in the project [20]. Generally speaking, a problem that is found late in the project is more expensive than if it was found early [6]. Therefore, we need fast feedback cycles that enable to react early to pitfalls during requirements engineering.

Some of these issues require specific domain knowledge to be uncovered. For example, it is very difficult to detect with automatic approaches whether a requirements specification is lacking necessary features.

However, other issues can be detected more easily: If a specification states that a sensor should work with *sufficient accuracy*, without detailing what sufficient means in the context, the specification is incomplete. 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* can cause misinterpretations and difficult consequences during the acceptance phase of a product.

Consequently, an approach that gives requirements engineers and project participants fast feedback on possible issues in the specification could provide valuable feedback. However, since requirements in industry are nearly exclusively written in natural language [21] and natural language has no formal semantics, these issues are hard to detect. To face the challenge of fast feedback and the imperfect knowledge of a specification's semantics, we created an approach that is based on what we call *requirements (bad) smells*, which are concrete symptoms for a requirement artefact's quality defect.

In this paper, we settle on the ISO/IEC/IEEE 29148:2011 standard [15] (in the following: *ISO 29148*) as a definition for requirements quality. The standard supplies a list of so-called *Requirements Language Criteria*, such as loopholes, ambiguous adverbs or comparative and negative statements. Based on this standard, we present a set of 8 smells that

indicate potential issues in requirements specifications according to the standard and implement an automatic smell detection for their discovery.

In two case studies, we applied the smell detection to 336 requirements and 53 comprehensive use cases from 9 specifications that were created in 2 different companies. Based on the results, we analyse with experts from the respective companies whether requirements smell analysis can be a beneficial approach for detecting defects in requirements specifications.

2. RELATED WORK

Various authors have worked on quality assurance of software requirements. Some focus on the classification of quality into characteristics [6], others develop comprehensive checklist, e.g. [19], [3], [2] or constructive approaches, e.g. [7], to name only a few.

Some researchers focussed on automatic detection of specific defects in requirements specifications. This includes detection of cloning in RE artefacts [16], detection of similar requirements [9], ambiguity [12], or detection of missing information and passive sentences [18].

Some groups have developed tools that focus on a broader understanding of requirements quality, instead of just a single aspect, e.g. the ARM tool [28] that is based on the IEEE 830 standard [14] that aims at developing metrics for requirements quality instead of giving feedback to developers. Consequently, only quantitative evaluation is performed.

Also the QuaRS tool [5, 8] analyses natural language requirements. However, their approach is based on a proprietary quality model and we could not find a discussion of the results with practitioners.

Circe [11] is able to detect more violations of quality characteristics in a more exact way by building logical models of the requirements specifications. However, their approach assumes that the specifications is written in certain patterns. This is often not the case in industry.

Research Gaps: We identified gaps from three sides: evaluation, quality definition and technique.

A major drawback that we see with the existing approaches, are the evaluations. Only few of the works apply their ideas on real industry specifications. Those who do apply their approach on real industry specifications, only give quantitative summaries, explaining which finding was detected and how often. Some authors also give examples of findings, but we could not find a detailed evaluation of how the findings relate to acknowledged requirement defects, i.e. together with the people who are supposed to use the approach. In our opinion, especially in the tacit domain of natural language, we must understand the impact of a finding in order to justify its detection.

Second, the existing approaches are based on proprietary definitions of quality, based on experience or on what can be measured. We have not seen an approach based on the novel ISO 29148 standard. Also, we have not seen an explicit understanding of how the produced findings relate to quality or quality defects. Our approach of smells covers this aspect systematically.

Lastly, from the technical side, different rules from the new quality standard required specific solutions: We have not seen the use of morphological analysis in requirements engineering quality assurance.

3. REQUIREMENTS SMELLS

In this section, we first introduce a short terminology on requirements smells, we then describe which smells we created, and finally explain how we detect the smells.

3.1 Requirements Smell Terminology

One concept for lightweight quality analysis are smells, which are proposed in the work by Fowler and Beck [10] to answer the question: At which point is the quality of code so low that we need to change it? According to the authors, the answer cannot be objectively measured, but we can only look for certain symptoms. This idea has also been transferred to (Unit) Test Smells [27] and finally to Natural Language Test Smells for user acceptance tests [13]. We apply the concept of smells to requirements.

Accordingly, we define *requirements quality* in terms of fitness-for-purpose, which implies that *bad quality* is a general property of a requirements artefact that has negative effects on activities in the software lifecycle.

Furthermore, a *quality defect* is a concrete instance or manifestation of bad quality in the artefact. This enables us to define a *requirements (bad) smell* as a concrete symptom for a requirement artefact's quality defect. In contrast to requirements defects, a requirements smell only shows a concrete indication for bad quality. Additionally, whether a smell turns into a problem is very *context*-specific. Lastly, we define a *finding* as instances of a smell, which might or might not be a defect.

3.2 Requirements Smell Design

We develop requirements smells based on an existing definition of quality. For this paper, we took the ISO 29148 requirements engineering standard [15] as a baseline. The reasons for this standard were two-fold:

First, the ISO 29148 standard has been created to harmonise a set of existing standards, including the well-known IEEE 830:1998 [14] 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, singularity etc. The standard furthermore describes the usage of requirements in different project phases and describes exemplary contents and structure for requirements specifications.
Therefore, we argue that this standard is based on a broad agreement and acceptance. Recent literature studies come to the same conclusion [24].

Second, the standard provides readers with a list of so-called *requirements language criteria*, which should help to choose proper language for requirements specifications. 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. [15, p.12]

In detail, the requirements language criteria consist of the following elements: [15]

Ambiguous Adverbs and Adjectives refer to adverbs and adjectives that are unspecific.
Examples: *almost always*, *significant*, *minimal*.

Vague Pronouns are unclear relations of a pronoun.
Example: *The system must have a keypad and a keyboard.* It *should use the German layout.*

Table 1: All implemented Smells

Smell Name	Implementation
Ambiguous Adverbs and Adjectives Smell	Dictionaries
Vague Pronouns Smell	POS tagging: Substituting pronouns
Subjective Language Smell	Dictionaries
Comparative Phrases Smell	Morph. Analysis: Adjectives and adverbs in comparative form POS tagging: Conjunctions of comparison
Superlatives Smell	Morph. Analysis: Adjectives and adverbs in superlative form
Negative statements	Dictionaries
Non-verifiable Terms Smell	Dictionaries
Loopholes Smell	Dictionaries
Incomplete references	Not implemented

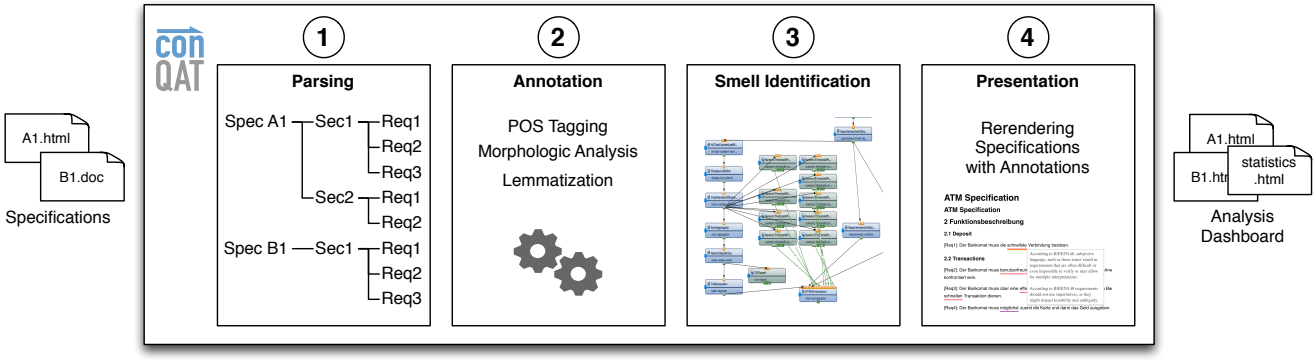

Figure 1: The Overall Smell Detection Process

Subjective Language refer to words of which the semantics is not objective.
Examples: *user friendly*, *easy to use*, *cost effective*

Comparative Phrases are used in requirements that express a relation of the system to specific other systems.
Examples: *better than*, *higher quality*

Superlatives are used in requirements that express a relation of the system to all other systems.
Examples: *best performance*, *lowest response time.*

Negative Statements are "statements of system capability not to be provided"[15]. Some argue that negative statements can lead to underspecification.
Example: *The system must not accept VISA credit cards.*
For this example, a more complete specification describes how the system reacts on the unaccepted input.

Open-ended, Non-verifiable Terms are hard to verify as they offer a choice of possibilities.
Examples: *provide support*, *but not limited to*, *as a minimum*

Loopholes enable stakeholders to ignore certain parts of the specification.
Examples: *if possible*, *as appropriate*, *as applicable*

Incomplete References are references that a reader cannot follow (e.g. no location provided).
Example: *[1] "Unknown white paper". Peter Miller.*

In the following we use all of these characteristics except for *incomplete references* as there were no explicit references in our specifications. All remaining features were considered as smells for bad quality of requirements specifications. At this point we assume that these criteria apply for all specifications; however, we will discuss the appropriateness of the given list based on concrete experience from the case studies in Section 5.2.

3.3 Requirements Smell Detection

The requirements smell detection, as presented in this paper, serves the automatic identification of requirements smells to support further manual quality assurance tasks (and potential corrections of requirements smells). In the following, we introduce the process for the automatic part of the approach, i.e. the detection of requirements smells.

The process consists of four steps (see Fig. 1): Parsing the specifications into single requirements, annotating the requirements with meta-information, detecting the requirements smells and finally creating a human-readable presentation of the findings, which are displayed integrated into the specification.

For the annotation and smell detection phase we employ various techniques, including techniques from Natural Language Processing (NLP)[17]. The smell detection is based on three different techniques. Tbl. 1 gives an overview of the techniques used for each individual smell.

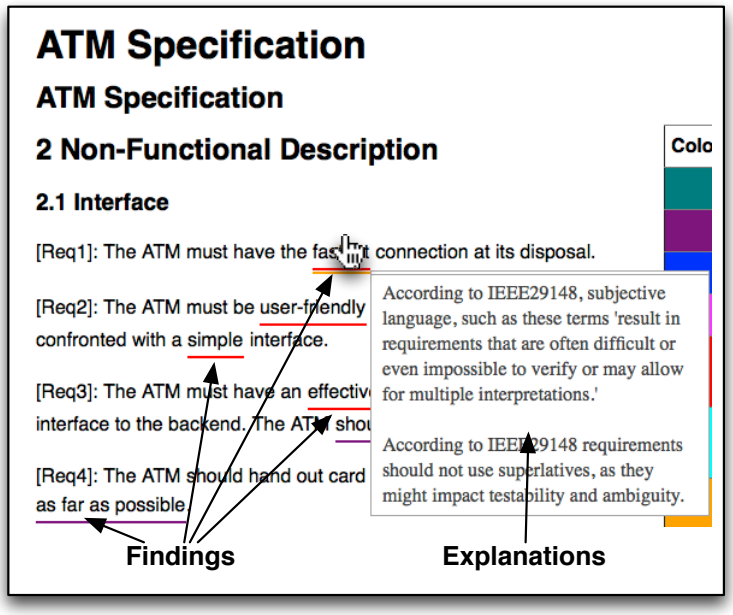

Figure 2: A Sample Output from the Smell Detection Tool for a Dummy Specification

POS Tagging: For two smells, we use a technique called part-of-speech (POS) tagging. Given a sentence in natural language text, it determines the role and function of each single word in the sentence. The output is usually a so-called *tag* for each word, e.g. whether a word is an adjective, a particle or a possessive pronoun. We used the Stanford NLP library [26] and the RFTagger [23] 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 [17]. 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 its inflection. This contains, among others to determine a verb's tense or an adjective's comparison. We use this technique to analyse if adjectives or adverbs are used in their comparative or superlative form.

Dictionaries: For the remaining five smells we use dictionaries, based on the proposals of the standard [15], and on our experience in the case studies. We furthermore apply lemmatisation for these words, which is a normalisation technique that reproduces the original form of a word. In other words, if a lemmatiser is applied to the words *were*, *is* or *are*, the lemmatiser will for all three return the word *be*. Lemmatisation is in its purpose very similar to stemming (e.g. the famous Porter Algorithm [22]), yet not based on heuristics, but on the POS tag as well as the word's morphological form.

The whole approach is implemented on top of the software quality analysis toolkit ConQAT[1]. ConQAT offers a platform for detailed data analysis, which we extended with NLP features. We furthermore developed a presentation that allows to read the finding in its context. In this presentation, the complete specification is displayed, and findings are annotated in a spelling-correction style. This follows the idea of smells as only indications that must be evaluated holistically in its context. Lastly, the system gives detailed information, when a user hovers a finding (see Fig. 2).

4. EVALUATION

We evaluated the approach in two case studies with 9 specifications from industry. In the following, we report on these studies.

4.1 Research Questions

For this first evaluation, we had three research questions in mind:

RQ1: Can we find defects with smell detection? First, we want to discuss the potential of the approach of requirements smells and lightweight smell analysis. To this end, we ask whether the approach produces results that pinpoint to defects of the requirements specification

RQ2: How many findings are present in the requirements specifications? Second, besides the potential of the approach, we also want to analyse the outcomes from a requirements engineering perspective. For this, we wanted to understand the distribution of findings across domains, specifications and the different smells.

[1] `http://www.conqat.org`

RQ3: Would requirements engineers use a requirements smell tool? Last, we wanted to have an opinion of both experts on whether or not a requirements smells approach would be useful for them.

4.2 Study Design

The study consisted of an automatic smell detection (see Fig. 1) and a manual evaluation phase (see Fig. 3):

In the smell detection phase, the specifications are first parsed from their original format (`.html` and `.doc`) into a machine readable format and parsed into individual requirements or use cases. Afterwards, each sentence and each word in each requirement is annotated with the information necessary for smell analysis, which is the POS information, morphologic information and the lemmatised word (see Sec. 3.3). Next, all smells detection algorithms are performed for each requirement, producing a set of findings for each requirement. These are subsequently presented in so-called *Analysis Dashboards*, which are human-readable presentations, including statistics and annotated views on the specification.

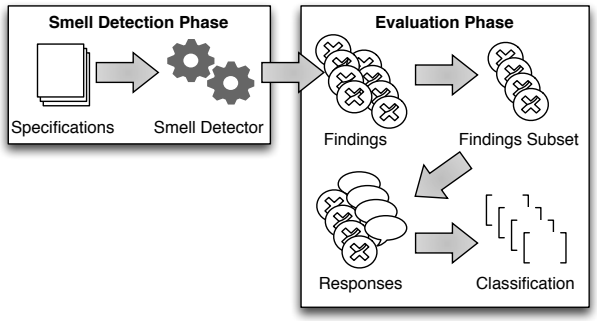

Figure 3: The Data Collection Process

In the evaluation phase, we first had to chose a subset of findings from the whole set of findings produced by the smell detection, as many findings are similar and the time of experts is rare. For the selection, we tried to filter false positives[2] and find samples from all smells.

For this paper, we were interested in the results of a lightweight static analysis of requirements. In order to receive unbiased and open opinions, we asked the participating experts of their opinion on these samples as an open question instead of a closed evaluations (e.g. ratings on a Likert scale).

Afterwards, two authors classified the qualitative answers independently using open coding (as known from the e.g. Grounded Theory approach [1]). Afterwards, we compared the codes as a validity procedure and in case of inconsistencies, resolved them.

Lastly, we asked the participating experts whether requirements smells tool could support their work.

4.3 Case & Subject Description

The case selection was driven by opportunity. We were approached by companies that wanted to understand whether there are issues in their requirements. All specifications were written in German language.

The two cases come from different domains and, thus, we have different representations of the requirements. Both companies are very successful and mature in their software development and are investing into good requirements engineering. In the following, we describe the chosen cases in detail.

Daimler AG: Daimler AG is one of the key players in the automotive industry with several hundred thousand employees and over 100 billion US dollar revenue.

At Daimler AG, we analysed six different specifications that were written by various authors. The specifications describe functionality in various 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 specifications A1-A6 that we analysed consist of 323 requirements (see Table 2). All of the specifications of Daimler AG analysed in our study were created by domain experts in a pilot phase after a change in requirements engineering at Daimler AG. We reviewed 22 findings with an external coach who works as a consultant for requirements engineering tightly collaborating with the group for many years.

Wacker Chemie AG: In the second case, we analysed specifications of business information systems from Wacker Chemie AG (or short: Wacker). Wacker is a globally active chemical company headquartered in Munich, Germany, with about 16,000 employees and a revenue of 4.63 billion Euros (2012). The systems that we analysed fulfil company-internal purposes, such as systems for access to Wacker buildings or support systems for document management.

We analysed three Wacker specifications that were written by five different authors. At Wacker Chemie AG, functional requirements are written as use cases (including fields for *Name*, *Description*, *Role*, and *Precondition*), whereas non-functional requirements are described in simple sentences. The specifications consisted of 53 use cases and 13 unstructured requirements (see Table 2). For the reviews of the findings, we selected 18 findings and discussed them with the Chief Software Architect, who also has several years of experience in quality assurance.

4.4 Results

We base our answer to RQ1 on the produced presentations of our tool and the responses to the findings by the experts, the answer to RQ2 on the total number of findings, and the answer to RQ3 on the written statements by the experts.

RQ1: Can we find defects with lightweight smell analysis?

We wanted to understand the potential of the approach in terms of ability to detect requirements defects. In the following, we first provide some examples before summarising the results from our analysis.

Examples: The automatic smell analysis produced 356 findings over all 9 specifications (see Tbl. 5), of which we selected 40 findings (∼11%) to discuss with the analysts in depth (see Fig. 3). An exemplary finding of each smell is shown in Tbl. 3. To demonstrate the types of issues that can be found, we will discuss three issues in depth here and go over a summary of the remainder afterwards.

[2]Obviously, this prevents us from analysing the precision of our smell detection.

Table 2: Study Objects

Specification	Topic	Size (Words)	Size (Sentences)	# Requirements	# Use Cases
A1	Adaptive valve control	2098	121	91	
A2	Exhaust control	2540	125	72	
A3	Driving information	215	13	12	
A4	Engine startup control	1118	76	44	
A5	Engine control	579	49	49	
A6	Powertrain communication	1248	66	55	
SUM		7798	450	323	
B1	Management of access control	2337	172	9	18
B2	Event notification	1162	103	3	19
B3	Document management	490	26	1	16
SUM		3989	301	13	53

Table 3: Exemplary Findings; shortened and translated from German by the Authors

Smell Name	Exemplary Finding
Ambiguous Adverbs and Adjectives Smell	If the (...) quality is **too low**, a fault must be written to the error memory.
Vague Pronouns Smell	The software must implement services for applications, **which** must communicate with controller applications deployed on other controllers. [Note: The translation is less ambiguous than the original finding in German, as the reflexive pronoun in English identifies its relation more clearly. The original requirement stated: *Die Software muss Dienste für Anwendungen implementieren, welche über ein Steuergerät hinaus mit anderen Steuergeräte-Anwendungen kommunizieren müssen.*]
Subjective Language Smell	The architecture as well as the programming must ensure a **simple** and **efficient** maintainability.
Comparative Phrases Smell	The display (...) contains the fields A, B and C, as well as **more exact** build infos.
Superlatives Smell	The system must provide the signal in the **highest** resolution that is desired by the signal customer.
Negative statements	One's own user **cannot** be deleted.
Non-verifiable Terms Smell	The system may only be activated, if all required sensors (...) work with **sufficient** measurement accuracy.
Loopholes Smell	**As far as possible**, inputs are checked for plausibility.

Subjective Language Smell In specification B2, a specification of a business information system, we found the requirement that the software *must ensure a simple and efficient maintainability.* This description of a non-functional requirement contains the classical error of violating verifiability. It is very hard to use this requirement properly in other activities of the software development lifecycle, e.g. engineers will find it hard to develop code against this requirement and testers will not be able to decide whether the resulting software fulfils the requirement during acceptance testing. The expert classified two instances of this finding as a major defect.

Vague Pronouns Smell The second example, found in specification A6, exemplarily shows the difficulties that come with complicated grammatical structures. In the example given in Tbl. 3, it remains unclear whether the *software*, the *services* or the *applications* should communicate with other applications. It is sometimes possible to deduce the reference from the context of the requirement, e.g. in this case we could take the word *other applications* as a hint that the word *which* refers to the word *application*. However, we still argue that the requirement contains potential for misunderstandings for all kinds of roles that are in contact with the specification.

Loopholes Smell The third example is taken from specification B1, a requirement artefact that describes the internal software system that manages the guests who access Wacker's properties, including chemical plants. The example contains the ambiguous phrase that a certain requirement should be fulfilled *as far as possible*. This is obviously problematic as, e.g. a developer might have a different opinion on the possibilities than the tester. Accordingly, testing will be performed subjectively.

After showing 40 findings to experts, we classified the qualitative responses to summarise the results. Tbl. 4 shows a summary of the answers, with responses where the experts would take action listed at the top and rather rejecting responses at the bottom part of the table.

Table 4: Qualitative Summary Smell Findings (Open Coding)

Classification (Code)	Occurrence	Explanation
Potential problem	8	This finding revealed a potential problem.
Needs review	6	This requirement needs a review.
Implicit knowledge	4	There is some implicit knowledge, which should be written down.
Missing reference	2	There should be a reference at this point.
Major defect	2	This is a big issue that must be addressed.
Refinement expected	6	While this is not an issue here, it must be further explained and refined at a different point.
No need for high quality	2	This could be problematic, but this part of the specification is not so important (e.g. information only, see Sec. 5)
Domain specialists knowledge	4	This finding seems problematic, but is clear to a domain expert.
No problem	4	This is not a problem here.
Finding wrong	1	The smell detection did not work properly.
Unsure whether a negative is a problem	1	It is unclear whether and why this formulation should be a problem (see Sec. 5)

Responses: We can see that there are many requirements in which experts directly proposed actions on this specification. Two times, which are two findings similar to the `Subjective Language Smell` example in Tbl. 3, the expert concluded that this is a major defect. The findings revealed especially issues with unstated information, i.e. *implicit knowledge*, and *missing references*. Six times, the expert explained that the finding probably pinpoints to a defect and that he would suggest a further review.

On the other side, experts classified that for some of the findings they would not take any further action. The most frequent cause was related to the subject under analysis: 8 times the experts told us that we were looking at the wrong spot, either because they said that this part is not really relevant in the specification[3] or because they stated that the requirements should be refined in a different artefact. These findings imply that a detailed understanding of purpose of the requirements is necessary to detect issues where they really matter. Another very common reason was that *domain specialists knowledge* is the reason for the finding, but there it was not seen necessary to make this knowledge explicit.

To summarise, we can see that the smell approach is able to detect requirements defects, as exemplified and validated with the experts.

RQ2: How many findings are present in the requirements specifications?

We wanted to understand in how far findings of the different smells are present in the specifications. Therefore, we analyse the distribution of findings across three dimensions: How are findings distributed across specifications, domains, and requirements smells? Tbl. 5 shows the total number of findings for all natural language criteria of the standard.

We see that nearly *all specifications* are subject to smells. The distribution varies between 0.3 and 0.62 findings per sentence, with specification B3 as an outlier, which we will discuss in the next paragraph. Obviously, the number of findings increases with the size of the specifications.

Furthermore, it is interesting to see that the total number of findings (see Tbl. 5) are quite similar in *both domains* (0.42 smells per sentence for the A1-A6 and 0.55 smells per sentence for B1-B3). One discrepancy that we looked at in more detail are the number of `loophole` findings. The reason for this was an extensive use of the German verb *soll*, which translates to *should* and is thus non-binding in contracts (in contrary to *shall*; cf. [4] or [15]). Hence, we see this certain error multiple times, especially in specification B3. Requirements authors at Daimler AG, in comparison, are taught to use the standard modalities where appropriate. This explains the discrepancy between the specifications. This is especially reflected in the variable `Smells per Sentence` in Tbl. 5.

The *distribution between the smells* varies strongly. Striking are the number of `negative statements` findings and `vague pronouns` findings. A selection of `negative statements` findings that we presented to our industry partners has lead to discussions with both experts on which we will report in detail in Sec. 5. For the `vague pronouns`, the reason lies in the implementation: As explained in Sec. 3.3, the smell detector suggests all substituting pronouns as findings. However, it turns out in the study that this is an overapproximation. Even though some of the sentences are indeed hard to understand (e.g. the example from Tbl. 3), very often it was very clear which word was substituted by the pronoun. For example, one automotive requirement from specification A4 constrained *The gear lever must be in Position P or N. This is not the case for (...).* In this case, even though the pronoun *this* is substituting, the reference is nevertheless quite clear from the context. Future work could include deeper linguistic dependency analysis of sentences, e.g. following the work of Smith [25].

RQ3: Would requirements engineers use a requirements smell tool?

After the analysis and interviews, we asked the expert of both Wacker and Daimler AG if they can comment whether

[3]Some parts of the specification were only considered to be further information and thus should not need to be of high quality.

Table 5: Quantitative Summary of Smell Findings

Specification	All Smells	Smells per Sentence	Negative Statements Smell	Superlative Requirements Smell	Comparative Requirements Smell	Subjective Language Smell	Loophole Smell	Non-verifiable Term Smell	Vague Pronouns Smell	Ambiguous Adverbs and Adjectives Smell
A1	45	0.37	11	7	7	4	2	0	13	1
A2	57	0.46	14	1	5	6	4	2	24	1
A3	8	0.62	2	0	0	0	0	0	6	0
A4	29	0.38	8	0	1	3	1	1	15	0
A5	20	0.41	5	0	0	0	0	1	14	0
A6	32	0.48	13	0	7	0	0	4	8	0
Sum	191	0.42	53	8	20	13	7	8	80	2
B1	100	0.58	20	6	7	5	18	1	43	0
B2	31	0.30	3	0	9	2	2	0	15	0
B3	34	1.31	0	1	1	0	21	0	10	1
Sum	165	0.55	23	7	17	7	41	1	68	1

or not they think the method is a helpful support. Their answers were:

Expert 1: I think that smells can help to analyse a specification. To use this correctly, the following aspects should be considered:

First, the people who need to write the specification, received training which gives the required performance criteria. Second, abstraction level's must be taken into account during smell detection process, since at higher abstractions level's different criteria can not be met (e.g., vague pronouns or subjective language).

Expert 2: 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 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.

Even though this is just anecdotal and, thus, subjective evidence, it forms a first external impression which encourages us to invest more effort into the development of requirements smells and analyse the approach in more depth.

4.5 Threats to Validity

We made four choices that could have had an impact the validity of the results.

First, the classification for RQ1 was performed by the authors. To address this threat, we performed triangulation of the classification between the first and the second author. For this, the second author of the paper conducted an independent classification of the responses to the 40 selected findings without being directly involved in the discussions with the industry participants. Out of 40 classifications, the recoding resulted in 6 out of 40 inconsistent classifications, which were directly resolved in discussions, and 18 out of 40 classifications that were chosen on a more coarse grained level; for instance, the second author selected the code "OK" for responses that indicated to findings being not classified as problems while the first author could reveal more fine-grained codes such as "Unsure whether a negative is a problem". Our interpretation of the independent classification result is, thus, that the results of the coding are sufficiently reliable.

Second, we selected the study objects by opportunity. These were the requirements for which we could get feedback from people with knowledge about the systems. To the best of our knowledge there are no benchmark requirements sets with proper information about its quality. However, we analysed requirements of 9 different systems, both from systems engineering and software engineering of traditional business information systems.

Third, we selected the set of findings that were discussed with industry experts not at random, but at our choice. This was done purposely in order to understand if the approach is generally able to find defects. Due to this decision, the number of issues presented in the results are not necessarily representative for the whole set of findings, i.e. no conclusions can be drawn about the quality of the smell detection approach in terms of precision or recall. All conclusions drawn in this paper respect this assumption.

Last, we selected the expert reviewers by availability. Even though they were familiar with the specifications, in future it would be best to ask not coaches nor architects, but the team members who use the requirements, e.g. testers or developers.

5. DISCUSSION

The study brought up several further questions that have strong implications on future research. Therefore, we discuss several of these aspects in more depth.

5.1 Smells for Rapid Requirements Analysis

In this paper, we analysed the usage of requirements smells in the context of rapid feedback for requirements analysis. In our context, we were able to show that it is possible to find relevant defects with automatic smell detection. The (automatic) smell detection took 59 seconds for A1 to A6 and 24 seconds for B1 to B3 in total. We consider this duration to be an effort reasonable to most types of projects.

However, it is important to note that a smell approach cannot substitute manual reviews or inspections. A requirements smell detection can only pinpoint to possible defects or common pitfalls. As a matter of fact, considering the low effort necessary to conduct an automatic smell detection, we believe that such a lightweight approach would greatly add to human inspections. Those manual approaches, in turn, can find deep flaws in requirements specifications. This includes violations of correctness as well as violations of completeness, such as missing functionality.

For this reason, we argue that requirements smells, just as in code smells, can serve as a very valuable input for inspections or reviews.

5.2 ISO 29148 Language Criteria as Smells

Industry experts did not agree on all natural language criteria that were proposed by ISO 29148.

This was especially the case for the `Negative Statements Smell`. The question whether or not findings of this smell lead to a problem was strongly discussed: On the one hand, the standard argues that negative phrases and statements should be avoided as a type of "unbounded or ambiguous terms". One could argue that formulating requirements in negative statements can lead to incompleteness. For the example given in Tbl. 3, the requirements specification lacks the information on how the system should react in case the user tries to delete his own dataset, or how else the system should prevent this to happen.

There are, on the other hand, non-functional requirements that are very hard to formulate in positive statements, e.g. requirements describing the prevention of system access. In any way, we need to find ways to understand and prove the impact of findings in a less argumentative (as proposed by the standard) and more empirically sound manner.

Another smell that produced interesting results was the `Superlative Requirements Smell`. The reasoning for this smell is that a requirements that is stated as, e.g. *highest resolution of a signal* or *fastest response of a sensor*, is inherently difficult to verify. However, it again depends on the context of the requirement. Taking the first example, if all possible resolutions are clear (i.e. if the set of possibilities is finite) the requirement is indeed verifiable. A similar argumentation holds for the `Comparative Requirements Smell`. We can see here, that the Requirements Smells and Requirements Smell Detection must be improved by adding a context to the smell definition. This improves the understanding on when a finding of a smell turns into a defect. It remains open, however, how to include this knowledge into a smell detector.

5.3 Implementation of Smell Detection

Besides understanding how to define the smells and understand which are the best smells to pinpoint to problematic spots in requirements artefacts, we can furthermore discuss how to detect the smells most appropriately after their definition. In this paper we basically made use of three techniques: Dictionaries, POS tagging and morphological analysis.

Some implementations depend on solely dictionaries (as others have done with similar problems before [3]). To our experience, this remains the most vague way of detecting certain smells as the dictionaries can only detect a finite set of words. While this is a perfect solution for smells, where there are only finite forms of this smell (also called *closed classes* in natural language processing [17]; e.g. there is only a small set of words that express negation, like "not", "neither", "no", etc.), it is inherently imprecise when it comes to open classes, e.g. `Non-verifiable terms`. Dictionaries furthermore inherently struggle when it comes to domain-specific language. So far, the only chance we see in this case is to taylor each smell based on feedback from the respective domain.

Other implementations are based on POS tagging and morphological analysis. Since the specifications were written in German and morphological analysis libraries are sparse for the German language, we had to combine various libraries, such as the Stanford NLP [26] with German NLP libraries [23]. We could identify some false positives that arise from the imprecision that comes with these libraries. This is especially the case when it comes to domain-specific terms and proper nouns.

Some implementations could be further refined, if more information from word dependencies would be used (see Sec. 4.4). For example, for the `Vague Pronouns Smell` we could evaluate the linguistic structure even further and try to detect whether there are multiple nouns that this pronoun could refer to.

5.4 Subject of Analysis

In this study we treated all parts of requirements specifications equally and assumed that each requirement had to be of highest quality, e.g. unambiguous, exact and verifiable. While this is true for many cases, experts opposed this assumption in some cases.

In an interview one of the experts told us that a certain finding was an issue on requirements level, but that in this spot it was ok, because this particular requirement was intended to be on a more abstract, goal-like level. This indicates that it is important to understand and taylor the smell detection to the abstraction level and granularity of the requirements (cf. [19]).

The same also holds for information texts and introductions. It is still completely unclear to which extent the quality of these parts of the specifications matter.

6. CONCLUSION

In this study, we proposed a light-weight approach to detect requirements smells. This approach is based on the natural language criteria of ISO 29148 and serves to rapidly detect requirements that violate certain RE principles. We furthermore developed an implementation that is able to detect these violations using part-of-speech (POS) tagging, morphological analysis and dictionaries.

We applied the approach in two case studies to 336 requirements and 53 use cases taken from 9 specifications that were created in 2 different companies. We discussed the results with industry experts and concluded that the approach is suitable to detect relevant defects in requirements. It cannot find all possible defects but experts judged it as a valuable input for requirements reviews. Furthermore, we saw that violations of the natural language criteria are present across domains and various specifications; however, the study also shows that we need further analyses to understand the impact of these violations.

Future work includes understanding negative statements in requirements, enhancement of the smell detection via dependency analysis and development of new requirements smells through interviews with testers and developers. We also want to understand the scalability of the approach: We are currently working on the analysis of a large business requirements specification of an industry partner.

Acknowledgments

We want to thank Daimler AG, especially Heike Frank, and Wacker Chemie AG for their support during the case studies, as well as Sebastian Eder, Maximilian Junker, Benedikt Hauptmann and the anonymous reviewers for their helpful and encouraging reviews.

7. REFERENCES

[1] S. Adolph, W. Hall, and P. Kruchten. Using Grounded Theory to study the Experience of Software Development. *Empirical Software Engineering*, 16(4), 2011.

[2] B. Anda and D. I. K. Sjøberg. Towards an inspection technique for use case models. In *Software Engineering and Knowledge Engineering (SEKE)*, 2002.

[3] D. Berry, A. Bucchiarone, and S. Gnesi. A new quality model for natural language requirements specifications. In *Requirements Engineering: Foundation for Software Quality (REFSQ)*, 2006.

[4] S. Bradner. Key words for use in rfcs to indicate requirement levels, 1997. RFC 2119.

[5] A. Bucchiarone, S. Gnesi, and P. Pierini. Quality Analysis of NL Requirements : An Industrial Case Study. In *Requirements Engineering*, 2005.

[6] A. Davis, S. Overmyer, K. Jordan, J. Caruso, F. Dandashi, A. Dinh, G. Kincaid, G. Ledeboer, P. Reynolds, P. Sitaram, A. Ta, and M. Theofanos. Identifying and measuring quality in a software requirements specification. In *Software Metrics Symposium*, 1993.

[7] C. Denger, D. M. Berry, and E. Kamsties. Higher quality requirements specifications through natural language patterns. In *Software Science, Technology, and Engineering*, 2003.

[8] 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 *NASA Goddard Software Engineering Workshop*, 2001.

[9] D. Falessi, I. C. Society, and G. Cantone. Empirical Principles and an Industrial Case Study in Retrieving Equivalent Requirements via Natural Language Processing Techniques. *Software Engineering*, 39(1), 2013.

[10] M. Fowler and K. Beck. *Refactoring: improving the design of existing code*. Addison-Wesley Professional, 1999.

[11] V. Gervasi and B. Nuseibeh. Lightweight validation of natural language requirements. *Software: Practice and Experience*, 32(2):113–133, Feb. 2002.

[12] B. Gleich, O. Creighton, and L. Kof. Ambiguity detection: Towards a tool explaining ambiguity sources. *Requirements Engineering*, 2010.

[13] B. Hauptmann, M. Junker, S. Eder, L. Heinemann, R. Vaas, and P. Braun. Hunting for smells in natural language tests. In *International Conference on Software Engineering (ICSE)*, 2013.

[14] IEEE Computer Society. IEEE Recommended Practice for Software Requirements Specifications. Technical report, 1998.

[15] ISO, IEC, and IEEE. ISO/IEC/IEEE 29148:2011. Technical report, ISO IEEE IEC, 2011.

[16] 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 *International Conference on Software Engineering (ICSE)*, 2010.

[17] D. Jurafsky and J. H. Martin. *Speech and Language Processing*. Pearson Education, 2014.

[18] L. Kof. Treatment of Passive Voice and Conjunctions in Use Case Documents. *Natural Language Processing and Information Systems*, 2007.

[19] A. V. Lamsweerde. *Requirements Engineering*. John Wiley & Sons, 2009.

[20] D. Méndez Fernández and S. Wagner. Naming the Pain in Requirements Engineering: Design of a Global Family of Surveys and First Results from Germany. In *Evaluation and Assessment in Software Engineering (EASE)*, 2013.

[21] L. Mich, F. Mariangela, and N. I. Pierluigi. Market research for requirements analysis using linguistic tools. *Requirements Engineering*, 9(1), 2004.

[22] M. Porter. An algorithm for suffix stripping. *Program: electronic library and information systems*, 1980.

[23] H. Schmid and F. Laws. Estimation of conditional probabilities with decision trees and an application to fine-grained POS tagging. In *Conference on Computational Linguistics*, 2008.

[24] F. Schneider and B. Berenbach. A Literature Survey on International Standards for Systems Requirements Engineering. In *Conference on Systems Engineering Research*, 2013.

[25] N. A. Smith. *Linguistic structure prediction*. Morgan & Claypool Publishers, 2011.

[26] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2003.

[27] A. van Deursen, L. Moonen, A. van den Bergh, and G. Kok. *Refactoring test code*. CWI, 2001.

[28] W. M. Wilson, L. H. Rosenberg, and L. E. Hyatt. Automated analysis of requirement specifications. In *International Conference on Software Engineering (ICSE)*, 1997.