Reviewing Natural Language Requirements with Requirements Smells – A Research Proposal –

Henning Femmer
Technische Universität München
Institute for Software & Systems Engineering
Munich, Germany
femmer@in.tum.de

ABSTRACT

The quality of requirements artifacts, such as software requirements specifications, is crucial for the success of a software development project, because the later a defect is found the more expensive it is to fix. However, as virtually all requirements are still written in natural language, and requirements artifacts grow often large, they are very hard to review for quality due to the imprecise nature of natural language.

In contrast, it is easier to not find quality, but symptoms of violations of quality, because they often leave concrete trace in the artifacts. For example, passive sentences in requirements are said to make testing harder as they can potentially hide the actor. Here it is easier to find the symptom of violation of testability, i.e. a passive sentence, than to prove that the requirement is "easily testable". This is a concept well known for code quality as code (bad) smells, which has been proposed by Fowler and Beck.

We suggest introducing the smell concept to requirements engineering in order to find possible violations of requirements quality. Consequently, a requirements (bad) smell is a concrete symptom for a requirement artifact's quality defect in the usage context of a certain activity.

The proposed research aims at understanding whether smells can help reviewing natural language requirement artifacts by pointing out to symptoms for potential quality defects in order to improve quality reviews of requirements.

Categories and Subject Descriptors

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

General Terms

Requirements Engineering, Quality Assurance, Natural Language

IDoESE '13 Baltimore, Maryland USA

Keywords

Requirements Engineering, Analytical Quality Assurance, Requirements Smells

1. INTRODUCTION

In a software development process of large scale, it is very important to create high quality requirements artifacts early on, because defects get more expensive the later they are found: If a defect in the requirements is discovered late, e.g. during system test, we must not only fix the requirements artifact, but also code, tests, etc. in order to support for consistency [4]. Various contributions show a up to 200:1 ratio of costs in detecting defects of requirements in the maintenance phase in contrast to the requirements phase [4].

Hence, industry requests for requirements quality assurance (QA) [20]. Due to the fact that most requirements artifacts today are written in natural language, QA often is performed through reviews[1]. However, as always when depending on reviews, inspecting large natural language documents is usually expensive and error-prone.

Also, when reviewing for quality, the goal is usually not to find high quality requirements, but low quality requirements that could potentially result in issues. In Fowler and Beck's book on refactoring [7], a similar problem is discussed regarding the quality of source code: At which point is the quality so low that we need to change it? According to the authors the answer is not objectively and easily measurable; instead, they propose to look for symptoms of bad quality. Our working hypothesis is that the same holds for requirements quality. The fact that there is no accepted RE quality metric in the community indicates that the quality of requirements is indeed still an unsolved topic. Focusing on finding requirements (bad) smells could provide a solution to the requirements quality problem.

1.1 Requirements (Bad) Smells

We define requirements (bad) smells as follows:
Definition: A *requirements (bad) smell* is a concrete symptom for a requirement artifact's quality defect in the usage context of a certain activity. In contrast to requirements defects, a requirements smell only shows a concrete indication

[1]We do not differentiate between different forms of reviews, such as Fagan inspections or equivalent. Our working hypothesis on this is that the results of the research apply for all different forms similarly.

for bad quality. Additionally, whether a smell turns into a problem is very context-specific.

Thus, a smell always adheres to the following smell model:

A requirements artifact or entity that is analyzed for the smell, e.g. use cases or scenarios. We use the terminology known from artifact-oriented requirements engineering [19].

An activity that is potentially affected by the smell (i.e. the impact of bad quality).

A context in which this smell is considered harmful.

Definition: We see requirements artifacts as means for a software engineering project instead of a root project goal[2]. *Quality* is hence defined as fitness-for-purpose, which implies that we define *bad quality* as a general property of a requirements artifact that has negative effects on activities in the software lifecycle. A *quality defect* is then a concrete instance or manifestation of bad quality in the artifact. This leads to a definition of quality as proposed by activity-based quality models, similar to [5]. Lastly, we define *findings* as instances of smells, which might or might not be a defect (see Fig. 1).

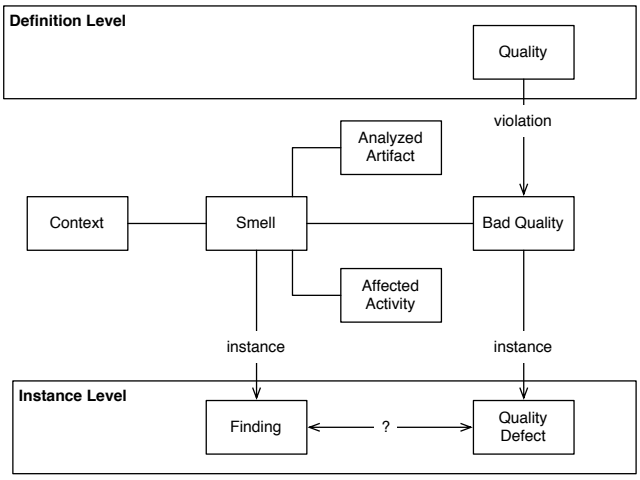

Figure 1: The Terminology

1.2 Example: Sentences in Passive Voice

A well-known example of a smell is a requirement that is written in passive voice. The smell could affect nearly all artifacts and, thus, it affects activities of the software lifecycle, because it makes testing and understanding a requirements sentence harder when it appears in a context where the actor is not entirely clear to all readers.

Yet, just as the impact of sentences in passive voice is debatable, the relation between these requirements smells and requirements quality remains unclear. This research aims at understanding which of these smells exist and how they affect requirements specifications.

Other examples for similar smells include subjective language, vague pronouns and superlative phrases.

[2]Unless the project's goal is to create a requirements artifact, obviously.

1.3 Preliminary Scope

Although the focus of our research is the investigation of requirements smells in general, we subsequently focus on artifact smells for natural language requirements. Hence, we do neither consider other representations, e.g. requirements in UML Syntax, nor do we look at requirements process smells, e.g. overly long discussions on reoccurring topics (unless they manifest themselves in the artifacts), even though these might be interesting next steps if the proposed research turns out to be fruitful.

2. RELATED WORK ON QUALITY OF REQUIREMENTS

Understanding and measuring quality in requirements engineering artifacts has been a long-time problem. For example, Dijkstra names consistency between requirements as a major problem at the NATO Software Engineering Conference in Garmisch in 1969 [21]. Consequently, textbooks such as [1] or [17] give concrete advice for establishing quality in requirements engineering artifacts, yet, it often comes in lists of best practices, such as the advice to avoid passive sentences or a check lists of common issues. These lists do not form a comprehensive picture of quality of requirements, nor do they justify why and how the proposals address the issue.

Hence, in the following we will first look at related work that focuses on quality from a top-down perspective and that describes which characteristics are defined for a high-quality requirements artifact. Afterwards we will describe existing bottom-up techniques for measuring and analyzing quality and conclude by summing up and identifying the gaps in research.

2.1 Defining Quality of Natural Language Requirements artifacts

Lindland et al. [18] sum up existing issues of requirements artifacts and derive three dimensions of quality: Syntactic quality (i.e. is the language used to express the requirements correct), semantic quality (i.e. is the system described by the requirements artifact valid and complete) and pragmatic quality (i.e. is the requirements artifact understandable by the audience). Krogstie et al. [16] extends this work, based on [22] by adding social quality as a level of agreement between stakeholders.

More recently, Katasonov et al. [14] defined 6 quality criteria for requirements artifacts and 8 criteria of requirements quality, based on two textbooks and a small number of issues from related work. Some of these criteria are discussed in more detail, it remains unclear, how to check these in detail and how important they are for a requirements artifact.

Schneider and Berenbach [23] sum up existing standards for requirements engineering. Here, most relevant is the IEEE830-1998 (IEEE Recommended Practice for Software Requirements Specifications) [11], which has since then been superseded by ISO/IEC/IEEE-29148 (Systems and software engineering – Life cycle processes – Requirements engineering) [12]. The IEEE-830 standard reduces quality of requirements engineering artifacts to 8 characteristics, such as correctness, completeness and traceability. ISO-29148 is far more exact in these characteristics and describes, inter alia, characteristics of the requirements artifact, characteristics of single requirements and also language criteria on how

requirements should not be formulated. For example, it is noted that certain wordings like superlatives often lead to problems with verifiability.

However, even though ISO-29148 is the most current and complete official standard on requirements engineering, assumptions like the one mentioned previously, are hardly discussed in academia. Therefore and because it is more detailed, we use the ISO-29148 as a first reference for requirements smells.

2.2 Smells and Metrics of Natural Language Requirements artifacts

Other authors approach the issue of quality of requirements artifacts by defining metrics.

First proposals by Davis et al. [4] describe 24 quality characteristics based on 16 existing papers. The resulting metrics for each of the quality characteristics are weighted in order to create a single metric for the quality of a requirements artifact. However, these metrics reveal several problems: Many metrics are either not measurable (e.g. correctness, as the authors properly analyze) or very simplified (e.g. modifiability is measured as presence of a table of contents and index), so that the construct validity is no longer given. Furthermore, weights are chosen deliberately and not validated.

A few contributions have been done on creating applicable metrics and indicators. E.g., Fabbrini et al. [6] and Buchiarone et al. [3] develop an approach based on 7 indicators. The studies presented quantify the number of findings, give some examples, but do not analyze in how far these indicators are able to detect violations of quality as suggested. Similar work has been done by Génova et al. [8] and Berry et al. [2], yet lacking an in-depth analysis of its abilities and drawbacks regarding to a holistic view on requirements quality.

Other authors assume the requirements artifact to have a certain structure in order to enable parsing it and building a light-weight formal model [9]. Especially rich in this regard is the area of ambiguity, a summary can be found for example in [15].

2.3 Discussion

In summary, the existing contributions either develop a top-down approach, resulting in defining metrics that are clearly not measurable, or defining a set of metrics or indicators by measuring "what you can measure". What is lacking, yet essential, is an understanding of the link between the top-down and the bottom-up approach, i.e. which quality characteristics can be analyzed with smells and where are other methods needed.

3. RESEARCH OBJECTIVE AND APPROACH

The research goal of the proposed work is to understand the possibilities of using requirements smells for supporting quality reviews of requirements artifacts.

In order to reach this goal, we need to answer three major research questions:

RQ1: Which requirements smells exist and how can they be classified?

RQ2: Which smells can be detected automatically?

RQ3: To which extent and under which conditions can requirements smells find quality defects?

3.1 RQ1: Which requirements smells exist and how can they be classified?

Understanding the current state of requirements smells has to start with building knowledge of how quality is currently understood in requirements engineering.

We need to understand which quality characteristics and which smells exist by reviewing standards, related research fields and practice. From these, we can derive a theory (or taxonomy) of smells.

Approach

We use the grounded theory approach as proposed by Glaser and Strauss [10] to create a theory of smells. Thereby, we can make use of various sources and build a theory as a set of hypotheses or falsified statements that describes the state-of-the-art. Furthermore, the theory is an important foundation for the next steps.

We apply the grounded theory approach on the standards ISO-29148-2011 [12] and IEEE-830-1998 [11], add information from a large industry partner, and validate it by observing requirements reviews in practice as well as through a survey.

The created theory consist of four categories and their relations (see Fig. 3):

- **Requirements smells** are the indicators that are supposed to direct towards quality issues, e.g. passive voice.
- **Requirements entities** are the artifacts or content items which might inhibit a smell, e.g. use cases, scenario steps, natural language requirements, etc.
- **Requirements activities** are the activities that are negatively affected by a smell, e.g. creating oracles for system tests.
- **Requirements quality characteristics** are the abstracted quality goals, e.g. testability.

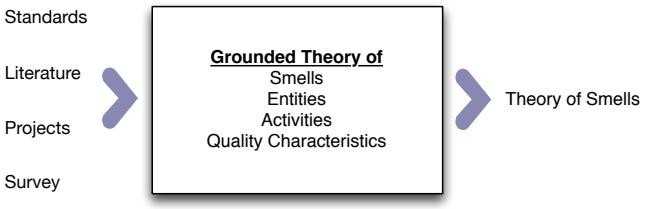

Figure 3: Research Question 1

With these categories we can use axial coding to understand which quality characteristics are connected to which smells. We will subsequently understand which quality characteristics can easily be analyzed and which cannot yet.

3.2 RQ2: Which smells can be detected automatically?

In the founding work by Fowler and Beck [7], smells are not connected to any kind of decision procedure. In contrary,

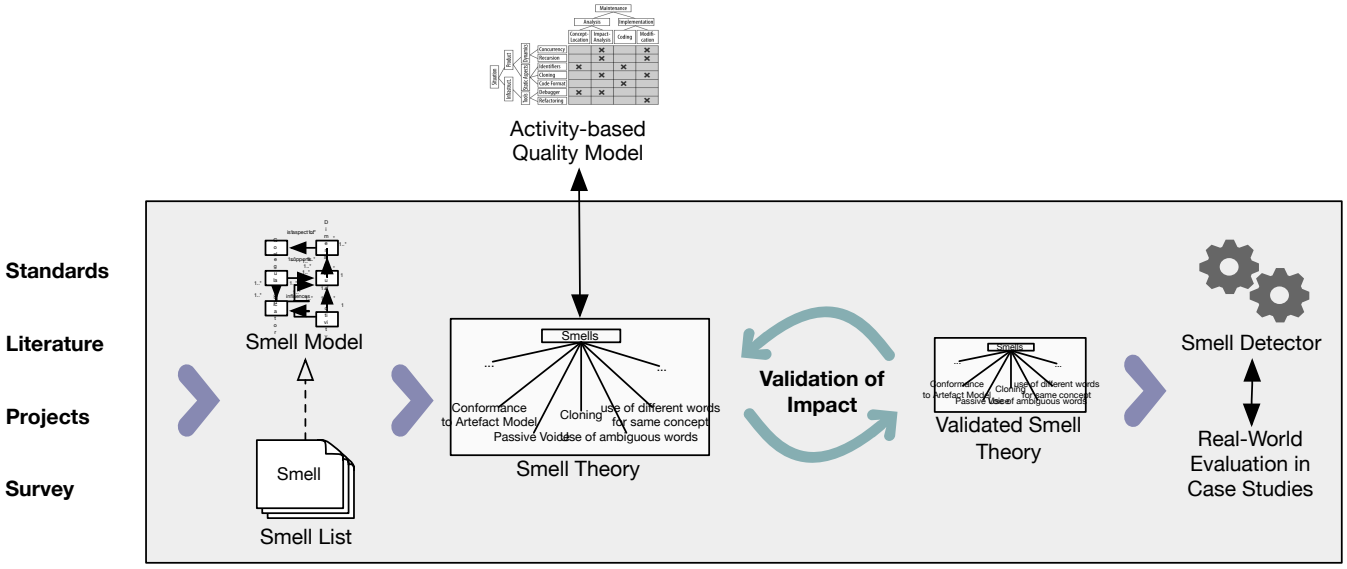

Figure 2: The proposed approach

smells are considered very subjective and imprecise. However, after their initial proposals, detection of various code smells have been automated, such as code clones [13] or god classes [24]. Also, most authors using the concept of smells focus on smells that can be automatically detected [25]. This might be due to the fact that the approach is most useful when the smell detection is cheap, as it is with automation.

Accordingly, one needs to understand which of the requirements smells defined in RQ1 can be automatically detected and which cannot. This includes the question of the internal validity of requirements smells, i.e. the question whether subjects find the same smells on a text or not.

However, it is important to note that also requirements smells that cannot be automatically detected might be of use during requirements quality analysis, as they might serve as supporting framework for reviews.

Approach

In order to understand whether smells can be detected automatically, we will rely on two sources: Related work from literature describes a few implementations of detecting certain aspects in requirements. Furthermore, for those smells not present in literature, we propose possible implementations, based on the open quality analysis framework ConQAT[3].

This approach will result in a more detailed theory as well as a prototype for a tool that can automatically detect smells in requirements and thus serves as an input for reviews.

Hypotheses

As this is a crucial step, we analyzed possible outcomes of this research question and discuss how we will proceed with this research based on the outcome. We see three possible results:

No requirements smells can be detected automatically. If it turns out that all smells need to be detected manually, we will accordingly focus on manual reviews and understand how smells as written descriptions impact reviews.

Some requirements smells can be detected automatically (working hypothesis). If we are only able to automate some smells, we will focus on these and elaborate our understanding of which characteristics of quality can be analyzed with smells (see RQ3).

All requirements smells can be detected automatically. At this stage we consider this very unrealistic. However, if this is the case, we will strongly focus on automation and performance of the automation in case studies.

3.3 RQ3: To which extent and under which conditions can requirements smells find quality defects?

The theory of RQ1 only depicts the current understanding of the area and the answers from RQ2 describe the possibilities and potential. It is, however, more important how the requirement smells really impact the requirements quality and the following project. As defined above, a quality defect is a violation of quality in terms of a negative impact on a certain activity in the software engineering life cycle.

This includes two questions: First, the theory needs to be validated in order to understand if requirements smells really indicate requirements quality defects or not. Accordingly, we must test if every smell can lead to a defect. Second, the theory must be checked against completeness, which is a question whether defects from all quality characteristics can be found in real world case studies. If they cannot be found, we must understand whether this is because these defects do not appear in real world examples or if there is no proper smell for it. This second question will answer whether a complete picture of quality of requirements can be created with requirements smells.

[3] `www.conqat.org`

Approach

To answer this research question, we validate the theory from RQ1 in case studies and experiments. In these we will focus on smells that are can be found by automation (based on the results from RQ2) as it allows a broader and hence more thorough analysis of specifications.

After creating a prototype for finding requirements smells, we analyze real world requirements specifications and discuss the findings with the authors of those specifications. To answer the questions, we measure the following metrics (see Fig. 4):

1. How many findings do we detect?
2. How many of these findings were quality defects?
3. How many of these were considered useful by the writer of the requirements artifact?
4. Which conditions (i.e. context) led to the answer in Question 3?

We determine answers to these questions through interviews and analysis of change requests. Afterwards we use reviews focusing on quality characteristics from RQ1, of which we could not find any defects. We accordingly update the theory from RQ1 and reiterate through all research questions.

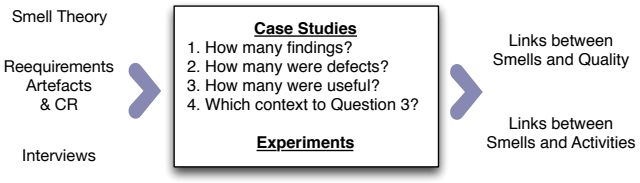

Figure 4: Research Question 3

3.4 Validity threats and their control

To the best of our knowledge, there are two major threats to validity in the proposed work.

Incorrectness and Incompleteness: First, the theory might be incorrect or incomplete. Inherent from the nature of analyzing qualitative data is the question when the necessary degree of saturation is reached. We thus plan to continuously validate our results in strong interaction with our industry partners to prevent that defects or smells are missing and to prevent that the theory describes defects or smells that are in fact no problem in practice.

Author's Bias: Second, we as the authors of the method, are biased in judging whether a smell really leads to a defect or not. Therefore, we plan to ask writers or users of the specifications for their judgment.

So far we see these two issues in the proposed work. However, as these are surely not the only existing threats we would be very interested in comments on threats that we did not recognize.

4. CURRENT STATUS

We are currently experimenting in each research question to prevent unexpected risks. To do so, we have derived the taxonomy depicted in Fig. 5 from ISO-29148. We are currently building a theory based on the standards and material from an industry partner and will incorporate indicators from related work next.

So far, the following three steps have been undertaken:

Smell Definition: To first understand the domain and it's terms properly, we have created (working) definitions for requirements smells and built a smell model that serves as a first reference for definitions (see Fig. 1).

Quality Definition: We are working on the foundations by building an activity-based requirements engineering quality model. As this is joint work in our group, and will not be part of the thesis, I omitted it from this research plan. This quality model will form the foundation of the requirements smells, as they are defined along a definition of quality. We plan to update this quality model as a group together with industry partners during upcoming projects.

Smell Implementation: We have implemented smells that analyze basic sentence structure, negative phrases, basic ambiguous words, legislative words and redundant passages and are currently working on superlatives and semantic text similarity. So far, these smells were created based on brainstorming ideas and ideas from the ISO-29148. The next step will be to create a more systematic theory based on the standard and structure the smells accordingly.

Case Studies: We are currently working on a case study with an industry partner. In this study, we are discussing the proposed work with requirements engineering coaches based on analysis of 5 different German requirements specifications.

One example for an issue, which we found by using our tool, is the following (translated into English by the authors):

> If the driver releases the clutch too quickly, the [feature] shall be deactivated.

This can be an issue as it remains unspecified what release speed is too fast and where it is ok. Accordingly, a tester who has to write a test case for this requirement, cannot test and decide whether this requirement is implemented or not.

The next step here is to use the newly implemented smells, and inspect the resulting findings together with the company to understand which of the findings are quality defects.

5. ISSUES

At our current stage, we see three issues:

Automation: We assume that the most potential for improvements are automatable requirements smells. However, in the original definitions, Fowler and Beck did not even plan to detect smells with tools (and for many of them there are still no tools [25]). Yet, smells might

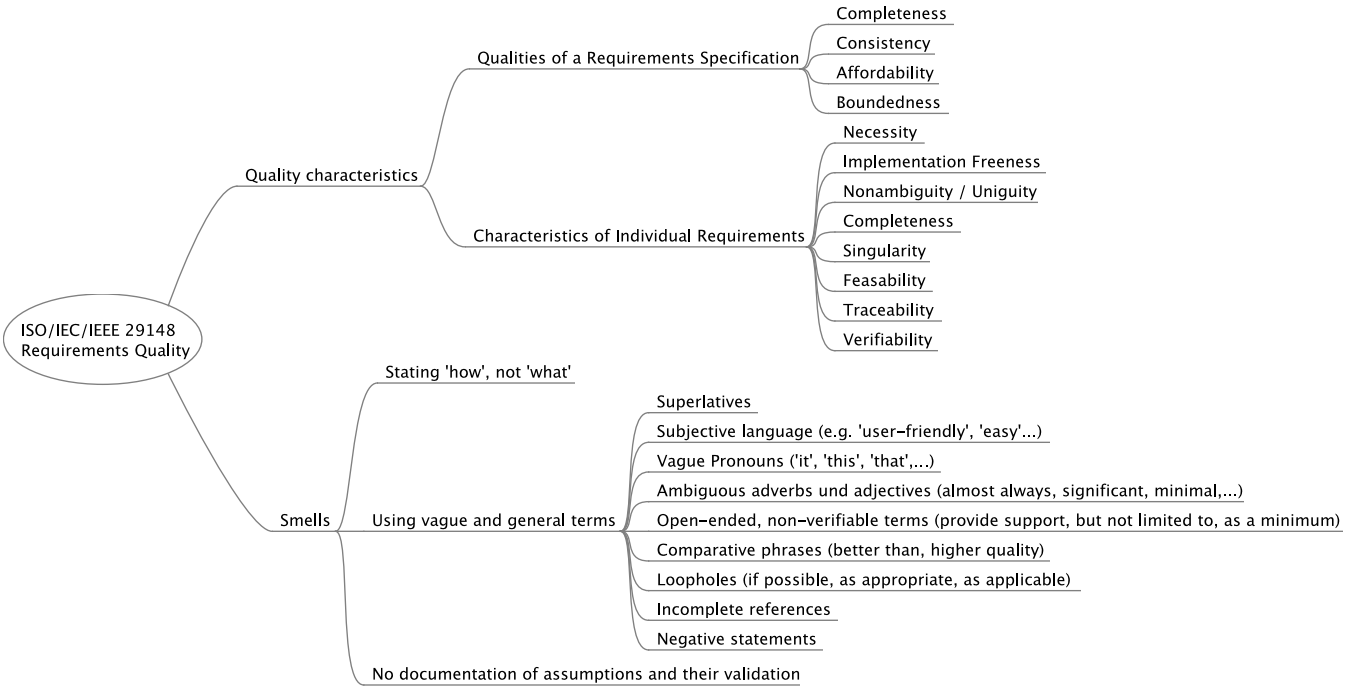

Figure 5: The taxonomy from ISO/IEE/IEC 29148

also help in reviews if they are not automated, could this be a risk to the research design?

Subjectivity: Some aspects of quality are very subjective, such as readability. How can we incorporate these subjective criteria into tools or quality model?

Validation of impact: We defined smells as heaving an impact on certain activities. Obviously, some of them can be validated in experiments. However, experiments on all impacts will be too time-consuming, so what is the most efficient way to validate the smells?

Risks of undetectable defects: We have a working assumption that we can find relevant issues with smells. However, there is a risk that especially automatically detectable smells only point to superficial problems, and do not lead to the underlying problems. Especially real shallow quality issues in requirements engineering, such as "we forgot to consider law XYZ", might indeed be unreachable for requirements smells. We are currently aware of this problem but are unsure whether this is a show-stopper, which we should consider in our research plan.

6. CONCLUSION

In summary, we propose to create a theory of requirements smells that is directly related to activities and quality characteristics. The theory is built on standards, as well as current research and industry state-of-the art. In the second step the theory is analyzed for automatability and validated in case studies and experiments to foster understanding, whether the perceived knowledge in the field really fits the existing links between requirements smells and requirements quality.

The outcome of the research has the potential to foster quality of requirements engineering both in academia and in practice. In academia, we target at a deeper understanding of quality in requirements artifacts. In practice, the goal is to improve quality reviews of natural language requirements.

Acknowledgements

I would like to thank Manfred Broy for his supervision of my thesis and the opportunity to hand this paper in at the doctoral symposium. Also, I would like to especially thank Daniel Méndez and Jakob Mund for the very valuable discussions and feedback on drafts on the paper.

7. REFERENCES

[1] I. F. Alexander and R. Stevens. *Writing Better Requirements*. Pearson Education, 2002.

[2] D. Berry, A. Bucchiarone, and S. Gnesi. A new quality model for natural language requirements specifications. In *Int'l Workshop on Requirements Engineering: Foundations of Software Quality (REFSQ)*, 2006.

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

[4] 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 *First International Software Metrics Symposium*, pages 141–152. IEEE Comput. Soc. Press, 1993.

[5] F. Deissenboeck, S. Wagner, M. Pizka, S. Teuchert, and J.-F. Girard. An activity-based quality model for maintainability. In *ICSM*, 2007.

[6] 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 *26th Annual NASA Goddard Software Engineering Workshop*, 2001.

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

[8] 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.

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

[10] B. G. Glaser and A. L. Strauss. *The discovery of grounded theory: Strategies for qualitative research*. Aldine de Gruyter, New York, 1967.

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

[12] ISO, IEC, and IEEE. IEEE29148. Technical report, ISO IEEE IEC, 2011.

[13] E. Juergens, F. Deissenboeck, B. Hummel, and S. Wagner. Do code clones matter? In *ICSE*, pages 485–495, 2009.

[14] A. Katasonov and M. Sakkinen. Requirements quality control: a unifying framework. *Requirements Engineering*, 11(1):42–57, Oct. 2005.

[15] 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*, 2008.

[16] J. Krogstie, O. Lindland, and G. Sindre. Towards a Deeper Understanding of Quality in Requirements Engineering. *Information Systems*, 1995.

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

[18] O. I. Lindland, G. Sindre, and A. Sø lvberg. Understanding Quality in Conceptual Modelling. *Ieee Software*, pages 42–49, 1994.

[19] D. Méndez Fernández, B. Penzenstadler, M. Kuhrmann, and M. Broy. A meta model for artefact-orientation: fundamentals and lessons learned in requirements engineering. In *Model Driven Engineering Languages and Systems*, pages 183–197, 2010.

[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 *EASE*, pages 183–194, 2013.

[21] Nato Science Committee. Software Engineering. Technical Report October 1968, 1969.

[22] K. Pohl. The three dimensions of requirements engineering: a framework and its applications. *Information Systems*, 19(3):243—-258, 1993.

[23] F. Schneider and B. Berenbach. A Literature Survey on International Standards for Systems Requirements Engineering. In *Conference on Systems Engineering Research*, volume 16, pages 796–805, Jan. 2013.

[24] J. Schumacher, N. Zazworka, F. Shull, C. Seaman, and M. Shaw. Building empirical support for automated code smell detection. In *ESEM*, page 1, New York, New York, USA, 2010. ACM Press.

[25] 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.