
CHARACTERIZING REQUIREMENTS SMELLS

Emanuele Gentili

MBDA Italy Spa, Rome, Italy
emanuele.gentili@mbda.it

Davide Falessi

University of Rome Tor Vergata, Rome, Italy
falessi@ing.uniroma2.it

ABSTRACT

Context: Software specifications are usually written in natural language and may suffer from imprecision, ambiguity, and other quality issues, called thereafter, requirement smells. Requirement smells can hinder the development of a project in many aspects, such as delays, reworks, and low customer satisfaction. From an industrial perspective, we want to focus our time and effort on identifying and preventing the requirement smells that are of high interest. **Aim:** This paper aims to characterise 12 requirements smells in terms of frequency, severity, and effects. **Method:** We interviewed ten experienced practitioners from different divisions of a large international company in the safety-critical domain called MBDA Italy Spa. **Results:** Our interview shows that the smell types perceived as most severe are Ambiguity and Verifiability, while as most frequent are Ambiguity and Complexity. We also provide a set of six lessons learnt about requirements smells, such as that effects of smells are expected to differ across smell types. **Conclusions:** Our results help to increase awareness about the importance of requirement smells. Our results pave the way for future empirical investigations, ranging from a survey confirming our findings to controlled experiments measuring the effect size of specific requirement smells.

Keywords Requirement smells · Requirement quality · Industrial case study.

1 Introduction

Software requirements specifications are usually written in natural language [19, 12] and may suffer from imprecision, ambiguity, and other quality issues, called thereafter, requirement smells [16]. Requirement smells can hinder the development of a project in many aspects, such as delays, reworks, and low customer satisfaction [1, 10].

Researchers identified many types of smells and developed mechanisms, such as tools or regular expressions, for smell identification [28, 18]. However, removing all smells is expensive given the high impact of the change on many artefacts such as design, code, testing, or certification [13]. Knowing which smell is important for whom, when, and why reasonably supports the reduction and prevention of smells. Therefore, it is key to gain insights into different types of smells.

Fernández et al. [8] reported a survey on the current status and issues in the requirements engineering process. We share with Fernández et al. [8] the need to gather additional empirical evidence about requirements and their quality. However, while they focus on the requirements engineering process, we focus on the requirements as artefacts written in natural language.

Montgomery et al. [18] reported a comprehensive mapping study on defining, improving, or evaluating requirements quality. However, to the best of our knowledge, no study investigated if and how the frequency, severity, or effects change across types of smells. Montgomery et al. [18] acts as our baseline to identify 12 types of smells: Ambiguity, Completeness, Consistency, Correctness, Complexity, Traceability, Reusability, Understandability, Redundancy, Verifiability, Relevancy, and Undefined.

We share the view of Femmer et al. [7] that "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." Thus, this research stems from the industrial need to focus our time and effort on removing and preventing the specific requirement smells that are of high interest.

The aim of this paper is to characterise 12 requirement smells in terms of frequency, severity, and effects. To the best of our knowledge, no previous study analysed how frequency, severity, or effects vary across requirement smells.

We interviewed ten experienced practitioners from different divisions of a large international company in the safety-critical domain called MBDA Italy Spa.

Our interview shows that the smell types perceived as most severe are Ambiguity and Verifiability while as most frequent are Ambiguity and Complexity. We also provide a set of six lessons learned about requirement smells such as that effects of smells are expected to differ across smell types.

The remainder of this paper is structured as follows. Section 2 discusses the related literature, focusing in particular on requirements and their smells. Section 3 reports the design, Section 4 the results. Finally, Section 5 concludes the paper and outlines directions for future work.

2 Related Work

A "requirement" refers to a specific functionality, constraint or quality that a system must possess in order to meet the needs of its users and stakeholders[19]. Since stakeholders' points of view may differ significantly from each other, and a common language is needed in order to communicate and share information among parties, requirements are usually expressed in natural language [19, 25, 20].

Kassab et al. [12] report that 61% of users prefer to express requirements in natural language, whereas only 33% use other semi-formal notations like UML. Despite the acquaintance a user can have with natural language, a non-systematic approach, i.e. unstructured, is likely to induce smells on the requirements specification, such as ambiguity, incompleteness, inconsistency and incorrectness [8]. These smells reasonably cause problems during the process stages and, lately, can determine the success or the failure of a project[1, 5].

Subramaniam et al. [26] and Mencl [17] propose approaches to prevent undesired effects of smells by means of limiting, i.e. structuring, the natural language syntax, to the extent of automatically generating use cases models for Object Oriented languages. Similarly, Femmer et al. [7] and Ferrari et al. [9] propose tools to automatically identify smells in requirements descriptions according to software requirements definition norms (like CENELEC EN 50128:2011) [9] or standards (like INCOSE or ISO 29184) [7]. These tools aim at driving the requirement elicitation process and assuring higher confidence in the requirement's quality.

In order to gain a deeper insight into Requirement Engineering state-of-the-art, Montgomery et al. [18] conducted a systematic mapping study on 105 relevant primary studies that use "empirical research to define, improve, or evaluate quality attributes". They identified 12 quality attribute themes, specified in 111 attributes sub-types, and reported that most of the studies concentrated on ambiguity, completeness, consistency and correctness quality attribute themes (63%). We share their quality attributes categorization and used them as categories for requirement smells.

2.1 Automated requirement smells detection

Requirement Engineering is acknowledged as an expensive, time-consuming, and error-prone process [2, 3]. This is especially true for complex systems with numerous requirements, e.g. thousand of requirements, making it challenging to obtain a comprehensive project specification overview. As a solution, automating smell detection becomes necessary in this context.

As reported by Montgomery et al. [18], a total of 41 distinct tools have been developed to detect requirement smells, and aspects such as ambiguity, incompleteness and inconsistency resulted as the most studied. For instance, Femmer et al. [7] focused on analyzing the syntax of requirements expressed in natural language, providing a tool, called *Smella*, implementing part of the Requirement Engineering standard ISO, IEC, and IEEE. ISO/IEC/IEEE 29148:2011 [11], and assessed the usefulness of the tool in the requirement elicitation process. Similarly, Seki et al. [22] designed a tool for detecting 22 "bad smells" (i.e. requirement smell sub-attributes) in use case description using structured natural language, achieving good performance in terms of precision and recall.

Veizaga et al. [28] developed a tool called *Paska* based on Rimay[27], and conducted an industrial case study on 13 system requirements specification documents from information system in financial domain, achieving a precision and recall in detecting smells of 89%. All the authors agree on the importance of assessing the usefulness of the developed tools through direct feedback from practitioners and on the necessity of a larger empirical evaluation.

2.2 Empirical evaluation

To the best of our understanding, the closest empirical evaluation to our study is [8]. If, on the one side, we share their need to gather more empirical data about requirements quality, on the other side, we differ in many aspects, such as the approach (survey vs interview) and the object under evaluation. Specifically, they focus on the requirements engineering process, whereas we focus on the requirements artefact as written in natural language. This allows us a way to create a preliminary knowledge base of which smells we should focus on the most.

Regarding the effectiveness of elicitation techniques, Davis et al. [4] conducted a systematic review, reporting that *interview* is the most commonly used elicitation technique, albeit there are no studies assessing that it is the most effective choice. Moreover, across interview strategies, the structured interview is the one gathering more information than unstructured interviews, sorting and ranking or thinking aloud techniques.

This turns out to be even more important if we consider the study conducted by [24], in which it emerges that companies with a "high-maturity rating", i.e. companies claiming to follow the best Requirement Engineering practices as a part of their quality management process, experience the same Requirement Engineering problems of companies with lower scores, remarking the necessity of looking deeply inside Requirement Engineering practices.

3 Methodology

In this section we report on the methodology we use in this work.

3.1 Industrial context

MBDA Spa is a multinational defence company specialising in the defensive and aerospace domain. We work closely with armed forces and defence organizations to provide advanced defence solutions. Our expertise lies in research, development, and integration of cutting-edge technologies to enhance national security and contribute to the defence capabilities of their client nations. The company comprises four national companies located in Italy, France, Germany, and United Kingdom. To conduct this study, we interviewed ten individuals from MBDA Italy Spa, which serves as the central site for software development supporting all the company's solutions.

3.2 Study design

In this work we use a qualitative semi-structured interview method, which has been proven to be a flexible instrument for investigating areas of interest whose boundaries are not clear nor complete [21]. Knowing how much a smell is frequent and severe reasonably supports reducing and preventing smells. To investigate which smell is particularly severe or frequent, we used the approach adopted by Fernández et al. [8]. Specifically, we asked about the three top and least severe smells and about the three top and least frequent smells.

Concerning types of smells, we use the categorization proposed by Montgomery et al. [18]: *Ambiguity, Completeness, Complexity, Consistency, Correctness, Traceability, Reusability, Understandability, Redundancy, Verifiability, Relevancy* and *Undefined*. We refer to Montgomery et al. [18] for their definitions.

Results about population and project characterization are reported in Table 1.

The list of questions is hereafter reported:

- **Interviewee characterization:**

1. What is your current role? (Role)
 - SW Engineer (SWEng): designs, develops, and maintains software systems for various applications.
 - Technical Expert (Tx): provides specialized knowledge and expertise in software requirements management and architecture modelling.
 - SW Project Leader (SPL): leads a software project, coordinates teams, manages resources, and ensures successful delivery of high-quality software solutions.
 - SW Group Leader (SGL): leads and coordinates a group of projects within a specific field, facilitating Software Project Leaders to ensure successful project execution and delivery.
 - Head of Department (HoD): leads the software department, setting strategic direction, manages SW Group Leaders and Project Leaders, and ensures efficient software development operations.
2. How many years of experience do you have in software development?(#YE)

Table 1: Population and project characterization.

Interviewees characteristics			Project characteristics					
Id	Role	#YE	#Req	#Dev	#LOC	#YP	#Exc	Domain
I1	SGL	25	1000	4	300K	3	1	SRT
I2	SGL	22	400	10	100K	2	2	SRT
I3	SPL	7	2000	12	250K	6	18	SRT
I4	Tx	21	300	7	250K	2	15	SRT
I5	SGL	21	750	7	70K	3	10	SRT
I6	SWEng	3	200	3	8K	5	1	HRT
I7	HoD/Tx	23	400	12	50K	4	20	SRT
I8	SWEng	18	2000	12	250K	6	18	SRT
I9	SGL	12	100	6	70k	5	1	HRT
I10	SWEng	16	2000	12	250k	6	18	SRT

- **Project characterization:** we asked information regarding projects that the interviewees are currently engaged in, or, if working on more than one, for a project they perceive as noteworthy in terms of requirement smells analysis.
 1. How many requirements does the project consist of?(#Req)
 2. How many developers does the team consist of?(#Dev)
 3. To which domain does the project belong to?(Domain)
 - Soft Real Time (SRT): refers to systems where meeting timing constraints is important but not critical. Occasional delays or missed deadlines may be tolerable as long as the overall system performance remains acceptable.
 - Hard Real Time (HRT): refers to systems where meeting strict timing constraints is crucial, and failure to do so can result in catastrophic consequences.
 4. How many software components, as the number of executables, does the project consist of?(#Exc)
 5. How many Line Of Code does the project consist of?(#LOC)
 6. How many years does the project last?(#YP)
- **Requirement smells:**
 1. What are the three most and least severe requirement smells?
 2. What are the three most and least frequent requirement smells?
 3. What are the effects of a certain requirement smell?
 4. Are there contexts in which the effects of a certain smell result mitigated/amplified?

3.3 Validity

Since the results rely on a small set of interviews from a single company, we recommend care in generalising results in other contexts.

There might be threats to validity even within our company. For instance, our population might not be representative of the company. To mitigate this threat, we selected subjects with a representative proportion of roles to face this threat.

Another possible threat to validity is selection bias, i.e., that the selected subjects might be biased towards specific answers. We believe this threat is negligible since we had a 100% acceptance rate and subjects with a representative proportion of roles.

An additional possible threat to validity is a wrong interpretation of subjects' answers. To face this threat, we provide no pressure on the time or direction of the answers. We also adopted a semi-structured interview protocol, which allowed us to spot possible misunderstandings while diving deep towards an answer. We also analysed the subjects' answers multiple times to ensure nothing was forgotten or misinterpreted.

4 Study Results

In the following we report the lessons learnt as extracted by analysing our interviews.

4.1 LL1: The perceived severity varies across types of smells.

Figure 1 reports the frequency distribution of the three most and least severe requirement smells. According to Figure 1, the perceived most severe smells are Ambiguity (80%), Verifiability (80%), and Consistency (60%). The perceived least severe smells are Relevancy (90%), Reusability (70%) and Redundancy (50%). Interestingly, only Completeness and Correctness have been identified as the most and least three severe by different interviewees.

4.2 LL2: The perceived severity of the same smell varies across project domains

We know that requirement smells can cause rework and time and cost overruns [1, 10], and in some cases, a smell might even be catastrophic. For instance, requirements concerning the performance of the system are key for HRT systems [15]. A single smell in a performance requirement of an HRT system might have a huge impact on the overall project or even people's lives. For instance, regarding Verifiability, "If a performance requirement does not come with clear time constraints, we have a huge verifiability problem. For instance, if a computation task is described with high priority and to be executed fast, this description does not lead to clear tests and therefore, the system might pass the test and eventually create system malfunction since the actual priority and speed constraints required by the production context differ from the tested ones." (cit. I9)

4.3 LL3: The perceived frequency varies across types of smells.

Figure 2 reports the frequency distribution of the most and least frequent requirement smells. According to Figure 2, the perceived most frequent smells are Ambiguity (70%), Complexity (70%) and Consistency (40%). The perceived least frequent smells are Understandability (60%), Reusability (40%), and Relevancy (50%). We note that differently from severity, the majority of smells are perceived as most and least frequent by different interviewees; this suggests less agreement among interviewees or, likely, the presence of other factors influencing the frequency of smells, such as roles or phases.

4.4 LL4: The frequency of a smell is perceived differently across roles or phases.

The requirements are managed over the development life cycles and get improved over the life cycle. Different companies have a proportion of different roles; each role gets the requirement at a different stage and hence at a different quality. Thus, some smells might not be frequent for a role because another role has already fixed the smell. For instance, "The presence of a SW Requirement Specification expert, who centralizes and pre-filters requirements affected by smells, is fundamental for reducing the frequency of such smells during the Development phase, and helps the whole Team to stay aligned with the given specification." (cit. I8)

4.5 LL5: The perceived effects may vary across types of smells.

We know that requirements smell can cause rework and time and cost overruns [1, 10]. Moreover, reasonably, specific smells cause specific problems. However, the specific effects of specific requirement smells, to the best of our knowledge, are unknown. Let's discuss the examples of the Verifiability, Ambiguity, and Complexity smells.

Regarding Verifiability, if it is unclear how to verify a requirement, then the verification might be incorrect and hence will likely require to be performed multiple times, thus impacting the time and cost of testing. An additional effect of the Verifiability smell is that bugs might not be found during testing, thus leading to decreased customer satisfaction and increased development costs. For instance, "The Verifiability of requirements can determine the success or the failure of a project: scarcely verifiable requirements determine special effort during the Coding phase (it is not clear how to code in order to provide evidence of the desired behaviour) and during the Testing phase (in fact the number and complexity of Test Cases grow significantly)... and a poorly tested SW is likely to exhibit bugs during the Maintenance phase, leading to a high impact in rework, time, extra costs and customer satisfaction, with a general loss of credibility of the Company." (cit. I2)

Regarding Ambiguity, if a requirement is unclear, this will likely need to go back and forth between requirements engineers and developers to identify and formalise a clear version of the content. Thus, an ambiguous requirement is the subject of many change requests. Specifically, "Across all the smells, Ambiguity is the one causing more problems: if evident, it can be addressed and solved at an early stage, before starting to develop code, with relatively little impact in terms of rework; but sometimes, it remains uncaught until Integration Test stage (or even worst, until Maintenance stage) causing bugs whose resolution will have, possibly, a very high impact in term of rework on all process stages, on costs and customer satisfaction." (cit. I2)

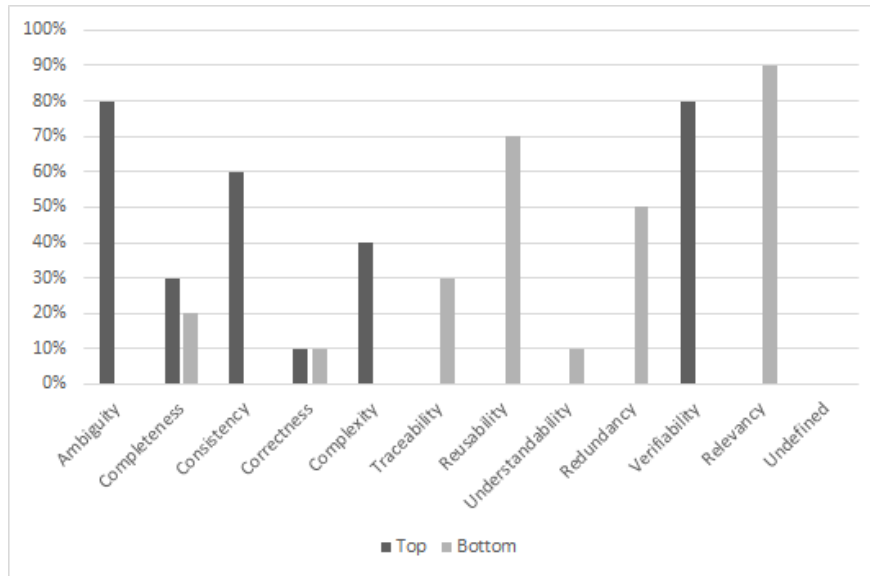


Figure 1: Distribution of the three most and least severe requirement smells.

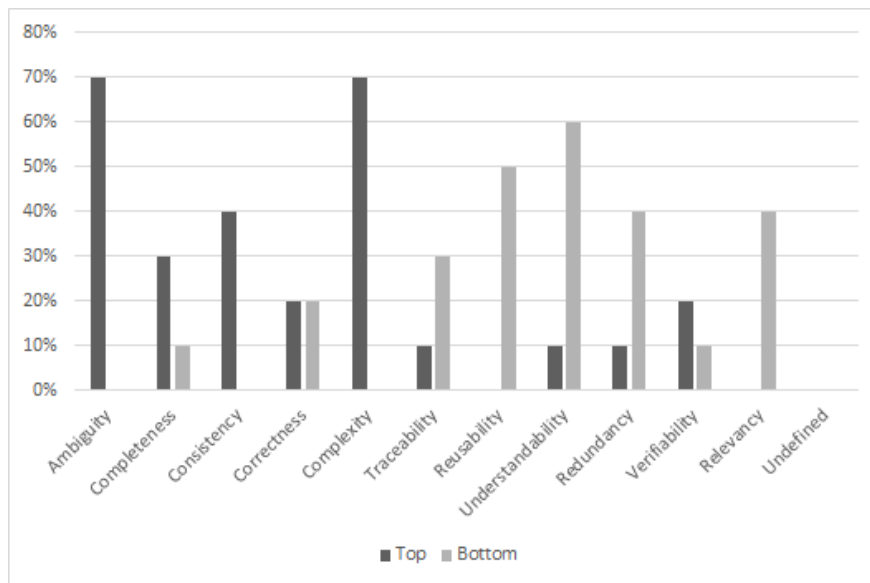


Figure 2: Distribution of the top three and bottom three requirement smell frequency.

Finally, regarding Complexity, it might be that a complex description of functionality leads to a complex implementation of that functionality. Specifically, "When a requirement is too Complex, one of the main effects is that practitioners tend to implement code with the same degree of complexity." (cit. I7)

We note that Verifiability and Ambiguity do not reasonably lead to complex code. Similarly, Complexity does not reasonably lead to decreased customer satisfaction. Thus, the impact of requirement smells is perceived as varying across smells.

4.6 LL6: The severity of a smell might change across the stage of the project.

We know that requirement smells can have negative effects on software development [1, 10]; however, we realize that the effects of smells might be null or even positive in some circumstances [7]. Let's have an example of how the effect of the underspecification[18], a sub-smell of the Completeness smell, changes across the development process stages and can even be positive. Regarding underspecification, we know from the literature that requirements get more specified over time as the clients get a better understanding of what they want [14]. Thus, some needs that are underspecified at the early stage of the project get specified over time; other needs might remain underspecified since the clients do not (need to) provide more details. Thus, the roles approaching an underspecified requirement at the early stage of the development process are in trouble since they need to make decisions based on assumptions that might change when clients will better specify their needs [6]. Counter-wise, roles approaching an underspecified requirement at a late stage are glad to have many options, knowing that no additional details will invalidate the chosen solution. Specifically, on the one side, "An underspecified functional requirements at an early stage can cause the SW architect to design an incorrect architecture with little-flexibly, not able to satisfy constraints that will come up later during the development stage. So it can potentially bring to the redesign of the whole architecture, at the price of losing time, money and increasing the frustration of the whole development team." (cit. I3) On the other side, "A too-abstract requirement is not necessarily bad news: from Project Leader and Technical Expert points of view, it provides a high degree of freedom in terms of selection of the most convenient software architecture, with the possibility to experiment with newest, and more adequate, SW solutions." (cit. I4) We do not know if this reasoning applies to requirements sub-smells other than underspecification.

5 Conclusions

From an industrial perspective, we want to focus our time and effort on identifying and preventing the requirement smells that are important. Knowing which smell is important for whom, when, and why reasonably supports the reduction and prevention of smells.

Our results rely on ten industrial experts and reveal that the smell types perceived as most important are Ambiguity and Verifiability, while as most frequent are Ambiguity and Complexity. We also provide a set of six lessons learned about requirement smells, such as that effects of smells are expected to differ across types. Our results help increase awareness about the importance of requirement smells.

To our best understanding, this study is the first attempt to characterise requirement smells in terms of severity, frequency and effects. Since the results rely on a small set of interviews from a single company, we recommend care in generalising results in other contexts.

We note that correlation does not imply causation. Smells perceived as important or related to effects are probably only correlated to rather than causing effects. The concept of inherent complexity is something we know for code smells [23] and likely applies to requirement smells too. Specifically, if something is easy to express in natural language, it is likely easy to design, code and test. Counter-wise, if something is complex, it remains complex regardless of how much time we spend improving its description. In other words, spending a lot of effort describing something complex might decrease the requirement smells, but it might not decrease the complexity of developing it. Of course, something easy might become complicated if described in a complex way. Thus, our results pave the way for future empirical investigations, ranging from a survey confirming our findings to mining software repositories to establish correlations between smells and effects on projects to controlled experiments measuring the size of smells effects.

References

- [1] Jarmo J Ahonen and Paula Savolainen. Software engineering projects may fail before they are started: Post-mortem analysis of five cancelled projects. *Journal of Systems and Software*, 83(11):2175–2187, 2010.
- [2] Barry W. Boehm. Software engineering economics. *IEEE Trans. Software Eng.*, 10(1):4–21, 1984. doi: 10.1109/TSE.1984.5010193. URL <https://doi.org/10.1109/TSE.1984.5010193>.
- [3] Alan M. Davis. *Software requirements - analysis and specification*. Prentice Hall, 1990. ISBN 978-0-13-824673-0.
- [4] Alan M. Davis, Óscar Dieste Tubío, Ann M. Hickey, Natalia Juristo Juzgado, and Ana María Moreno. Effectiveness of requirements elicitation techniques: Empirical results derived from a systematic review. In *14th IEEE International Conference on Requirements Engineering (RE 2006), 11-15 September 2006, Minneapolis/St. Paul, Minnesota, USA*, pages 176–185. IEEE Computer Society, 2006. doi: 10.1109/RE.2006.17. URL <https://doi.org/10.1109/RE.2006.17>.

- [5] Jeremy Dick, M. Elizabeth C. Hull, and Ken Jackson. *Requirements Engineering, 4th Edition*. Springer, 2017. ISBN 978-3-319-61072-6. doi: 10.1007/978-3-319-61073-3. URL <https://doi.org/10.1007/978-3-319-61073-3>.
- [6] Davide Falessi, Giovanni Cantone, Rick Kazman, and Philippe Kruchten. Decision-making techniques for software architecture design: A comparative survey. *ACM Comput. Surv.*, 43(4):33:1–33:28, 2011. doi: 10.1145/1978802.1978812. URL <https://doi.org/10.1145/1978802.1978812>.
- [7] Henning Femmer, Daniel Méndez Fernández, Stefan Wagner, and Sebastian Eder. Rapid quality assurance with requirements smells. *CoRR*, abs/1611.08847, 2016. URL <http://arxiv.org/abs/1611.08847>.
- [8] Daniel Méndez Fernández, Stefan Wagner, Marcos Kalinowski, Michael Felderer, Priscilla Mafra, Antonio Vetrò, Tayana Conte, Marie-Therese Christiansson, Des Greer, Casper Lassenius, Tomi Männistö, M. Nayabi, Markku Oivo, Birgit Penzenstadler, Dietmar Pfahl, Rafael Prikladnicki, Günther Ruhe, André Schekelmann, Sagar Sen, Rodrigo O. Spínola, Ahmet Tuzcu, Jose Luis de la Vara, and Roel J. Wieringa. Naming the pain in requirements engineering - contemporary problems, causes, and effects in practice. *Empir. Softw. Eng.*, 22(5):2298–2338, 2017. doi: 10.1007/s10664-016-9451-7. URL <https://doi.org/10.1007/s10664-016-9451-7>.
- [9] Alessio Ferrari, Gloria Gori, Benedetta Rosadini, Iacopo Trotta, Stefano Bacherini, Alessandro Fantechi, and Stefania Gnesi. Detecting requirements defects with NLP patterns: an industrial experience in the railway domain. *Empir. Softw. Eng.*, 23(6):3684–3733, 2018. doi: 10.1007/s10664-018-9596-7. URL <https://doi.org/10.1007/s10664-018-9596-7>.
- [10] M. Elizabeth C. Hull, Ken Jackson, and Jeremy Dick, editors. *Requirements Engineering, Third Edition*. Springer, 2011. ISBN 978-1-8499-6404-3. doi: 10.1007/978-1-84996-405-0. URL <https://doi.org/10.1007/978-1-84996-405-0>.
- [11] IEC ISO. Ieee. 29148: 2011-systems and software engineering-requirements engineering. Technical report, Technical report, 2011.
- [12] Mohamad Kassab, Colin Neill, and Phillip Laplante. State of practice in requirements engineering: contemporary data. *Innovations in Systems and Software Engineering*, 10:235–241, 2014.
- [13] Maria Kretsou, Elvira-Maria Arvanitou, Apostolos Ampatzoglou, Ignatios S. Deligiannis, and Vassilis C. Geroiannis. Change impact analysis: A systematic mapping study. *J. Syst. Softw.*, 174:110892, 2021. doi: 10.1016/j.jss.2020.110892. URL <https://doi.org/10.1016/j.jss.2020.110892>.
- [14] Philippe Kruchten. What do software architects really do? *J. Syst. Softw.*, 81(12):2413–2416, 2008. doi: 10.1016/j.jss.2008.08.025. URL <https://doi.org/10.1016/j.jss.2008.08.025>.
- [15] Phillip A Laplante et al. *Real-time systems design and analysis*. Wiley New York, 2004.
- [16] Alistair Mavin and Philip Wilkinson. Big ears (the return of" easy approach to requirements engineering"). In *2010 18th IEEE International Requirements Engineering Conference*, pages 277–282. IEEE, 2010.
- [17] Vladimir Mencl. *Deriving behavior specifications from textual use cases*. Citeseer, 2004.
- [18] Lloyd Montgomery, Davide Fucci, Abir Bouraffa, Lisa Scholz, and Walid Maalej. Empirical research on requirements quality: a systematic mapping study. In Gregor Engels, Regina Hebig, and Matthias Tichy, editors, *Software Engineering 2023, Fachtagung des GI-Fachbereichs Softwaretechnik, 20.-24. Februar 2023, Paderborn*, volume P-332 of *LNI*, pages 91–92. Gesellschaft für Informatik e.V., 2023. URL <https://dl.gi.de/20.500.12116/40098>.
- [19] Klaus Pohl. *Requirements Engineering - Fundamentals, Principles, and Techniques*. Springer, 2010. ISBN 978-3-642-12577-5. URL <http://www.springer.com/computer/swe/book/978-3-642-12577-5?changeHeader>.
- [20] Suzanne Robertson and James Robertson. *Mastering the requirements process: Getting requirements right*. Addison-wesley, 2012.
- [21] C Robson. *Real world research blackwell. 2° edição*, 2002.
- [22] Yotaro Seki, Shinpei Hayashi, and Motoshi Saeki. Detecting bad smells in use case descriptions. In Daniela E. Damian, Anna Perini, and Seok-Won Lee, editors, *27th IEEE International Requirements Engineering Conference, RE 2019, Jeju Island, Korea (South), September 23-27, 2019*, pages 98–108. IEEE, 2019. doi: 10.1109/RE.2019.00021. URL <https://doi.org/10.1109/RE.2019.00021>.
- [23] Dag I. K. Sjøberg, Aiko Fallas Yamashita, Bente Cecilie Dahlum Anda, Audris Mockus, and Tore Dybå. Quantifying the effect of code smells on maintenance effort. *IEEE Trans. Software Eng.*, 39(8):1144–1156, 2013. doi: 10.1109/TSE.2012.89. URL <https://doi.org/10.1109/TSE.2012.89>.

- [24] Badariah Solemon, Shamsul Sahibuddin, and Abdul Azim Abdul Ghani. Requirements engineering problems and practices in software companies: An industrial survey. In Dominik Slezak, Tai-Hoon Kim, Kiumi Akingbehin, Tao Jiang, June M. Verner, and Silvia Abrahão, editors, *Advances in Software Engineering - International Conference on Advanced Software Engineering and Its Applications, ASEA 2009 Held as Part of the Future Generation Information Technology Conference, FGIT 2009, Jeju Island, Korea, December 10-12, 2009. Proceedings*, volume 59 of *Communications in Computer and Information Science*, pages 70–77. Springer, 2009. doi: 10.1007/978-3-642-10619-4_9. URL https://doi.org/10.1007/978-3-642-10619-4_9.
- [25] Iain Sommerville and Peter Sawyer. *Requirements engineering: a good practice guide*. John Wiley & Sons, Inc., 1997.
- [26] Kalaivani Subramaniam, Dong Liu, Behrouz Homayoun Far, and Armin Eberlein. UCDA: use case driven development assistant tool for class model generation. In Frank Maurer and Günther Ruhe, editors, *Proceedings of the Sixteenth International Conference on Software Engineering & Knowledge Engineering (SEKE'2004), Banff, Alberta, Canada, June 20-24, 2004*, pages 324–329, 2004.
- [27] Alvaro Veizaga, Mauricio Alférez, Damiano Torre, Mehrdad Sabetzadeh, and Lionel C. Briand. On systematically building a controlled natural language for functional requirements. *Empir. Softw. Eng.*, 26(4):79, 2021. doi: 10.1007/s10664-021-09956-6. URL <https://doi.org/10.1007/s10664-021-09956-6>.
- [28] Alvaro Veizaga, Seung Yeob Shin, and Lionel C. Briand. Automated smell detection and recommendation in natural language requirements. *CoRR*, abs/2305.07097, 2023. doi: 10.48550/arXiv.2305.07097. URL <https://doi.org/10.48550/arXiv.2305.07097>.