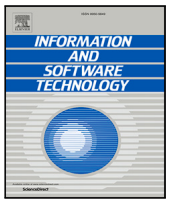
Information and Software Technology 187 (2025) 107823

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof



Practitioners' perceptions on requirements smells

Emanuele Gentili [a,b], Davide Falessi [b,*]

[a] MBDA Italy S.p.a., Via Monteflavio, 45, Rome, 00132, Lazio, Italy
[b] University of Rome Tor Vergata, Via del Politecnico, 1, Rome, 00132, Lazio, Italy

ARTICLE INFO

Keywords:
Requirement smells
Requirement quality
Industrial case study

ABSTRACT

Context: Software specifications are usually written in natural language and may suffer from imprecision, ambiguity, and other quality issues, hereafter referred to as requirement smells. Requirement smells can hinder project development 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 of high interest. We also want to identify the metrics to measure the effect of smells on a software project.

Objective: We aim to characterize types of 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 types of smells.

Methods: We interview ten experienced practitioners from different divisions of a large international company in the safety–critical domain called MBDA Italy Spa. Then we survey 58 people from the same company to support our findings and extend the analysis to metrics for measuring specific types of requirements smells effects.

Results: Our results show that the smell types perceived as most severe are Ambiguity and Unverifiability, while the most frequent are Ambiguity and Incompleteness. We also provide six Findings about requirements smells, such as that the effects of smells are expected to differ across smell types and stages of the project. our study suggests that measuring the effects of requirement smells may necessitate type-specific metrics.

Conclusion: Our results contribute to a greater understanding of the importance of addressing requirement smells and provide actionable insights for improving requirement quality in industrial settings. Our results pave the way for future empirical investigations, such as mining project repositories, to measure the specific effect type and size of specific requirements' smells.

1. Introduction

Software requirements specifications are typically written in natural language [1,2] and often suffer from imprecision, ambiguity, and other quality issues, collectively referred to as requirement smells [3]. These smells can significantly hinder project development, leading to delays, rework, and low customer satisfaction [4,5].

Researchers have identified various types of smells and developed mechanisms such as tools and regular expressions to detect them [6,7]. However, eliminating all smells is costly due to the widespread impact of changes on artefacts like design, code, testing, and certification [8]. Prioritizing which smells matter most based on context, stakeholders, and timing—can help reduce and prevent them effectively, making it crucial to understand different types of smells.

Fernández et al. [9] conducted a survey on the current state and challenges of the requirements engineering process. While we align with their call for more empirical evidence on requirements quality, their focus on the requirements engineering process contrasts with our emphasis on requirements as artefacts written in natural language.

Similarly, Montgomery et al. [7] presented a comprehensive mapping study on defining, improving, and evaluating requirements quality. Yet, to the best of our knowledge, no study has explored how the frequency, severity, or effects of smells vary across different types. We share the perspective of Femmer et al. [10] 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". This research responds to the industrial need to focus on identifying and mitigating the most critical requirement smells.

Our research question is: *How do the different types of requirement smells differ in an industrial context?* Thus, the aim of this paper is to

* Corresponding author.
E-mail addresses: emanuele.gentili@mbda.it (E. Gentili), falessi@ing.uniroma2.it (D. Falessi).

https://doi.org/10.1016/j.infsof.2025.107823
Received 23 May 2024; Received in revised form 6 June 2025; Accepted 16 June 2025
Available online 5 July 2025

characterize 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. This paper extends the preliminary study of Gentili and Falessi [11], which had the exploratory aim of uncovering findings about frequency, severity, and effects of requirements smells by performing interviews on 10 employees of MBDA Italy. In the current paper, we answer our research question by supporting and extending the six findings of the preliminary study via a survey of 58 people from the same company Our findings indicate that the smell types perceived as most severe are Ambiguity and Unverifiability, whereas the most frequent are Ambiguity and Incompleteness. We also provide a set of six findings about requirement smells such as that effects of smells are expected to differ across smell types.

This paper is organized as follows: Section 2 reviews relevant literature, with a particular focus on requirements and their associated smells. Section 3 details the research design, followed by the presentation of results in Section 4. Section 5 analyzes the findings and their implications. Section 6 discusses potential threats to the validity of this study. Finally, Section 7 concludes the paper and proposes directions for future research.

2. Background and related work

Requirements refer to the specific functionalities, constraints, or qualities a system must possess to meet user and stakeholder needs [1]. 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 [1,12,13].

Kassab et al. [2] 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 [9]. These smells reasonably cause problems during the process stages and, lately, can determine the success or the failure of a project [4,14].

Requirement quality has long been a focal point in Requirements Engineering (RE), with various studies emphasizing the importance of characteristics such as clarity, consistency, feasibility, and completeness. The IEEE Standard 29148:2018 [15] defines high-quality requirements as those that are clear, correct, and verifiable, offering precise guidelines to minimize misunderstandings and ensure successful implementation. Prior to the introduction of the term "requirement smell", researchers extensively investigated issues that affect requirement quality, identifying recurring problems like vagueness, inconsistency, and incompleteness [5]. These studies highlight the detrimental impact of poor-quality requirements on the entire software development lifecycle, laying a foundation for later work that formalized and categorized such issues under the umbrella of "requirement smells".

In order to gain a deeper insight into Requirements Engineering state-of-the-art, Montgomery et al. [7] 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 use them as categories for requirement smells.

Similarly, the INCOSE (International Council on Systems Engineering) standard [16] is a globally recognized framework that provides comprehensive guidelines for systems engineering processes, including the elicitation, specification, and management of requirements. In the context of software requirement analysis, the INCOSE standard emphasizes the importance of clarity, consistency, and feasibility in requirements, aligning closely with the detection and mitigation of requirement smells. Requirement smells, such as ambiguity, inconsistency, and redundancy, directly contradict the best practices outlined in the INCOSE standard, which seeks to minimize risks in system development by promoting precise and verifiable requirements. This makes the INCOSE standard an essential reference for detecting and addressing requirement smells, as adhering to its principles ensures higher-quality requirements that are less prone to interpretation errors and implementation challenges. We share their principles, and use them to identify requirement smells. To build on the work of Montgomery et al. [7], we mapped the quality attribute themes they identified to the requirement smells defined by the INCOSE standard [17]. Table 1 illustrates this correspondence, offering a (possible) linkage between the two categorizations to support systematic requirement smell detection.

Table 1
Interview to Survey categorization mapping.

Interview (Montgomery et al. [7])	Survey (INCOSE [17])
1:Ambiguous	Ambiguous:1, 8
2:Incomplete	Incomplete:2
3:Complex	Plural:3, 7
4:Inconsistent	Inconsistent:4
5:Incorrect	Incorrect:5
6:Non-Traceable	Infeasible:1, 5
7:Non-Reusable	Non-Conforming:5, 6, 8, 9
8:Non-Understandable	Inappropriate:5
9:Redundancy	Unverifiable:10, 6
10:Unverifiable	Unnecessary:9, 11
11:Non-Relevant	
12:Undefined	

Subramaniam et al. [18] and Mencl [19] 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. [10] and Ferrari et al. [20] propose tools to automatically identify smells in requirements descriptions according to software requirements definition norms (like CENELEC EN 50128:2011) [20] or standards (like INCOSE or ISO 29184) [10]. These tools aim to drive the requirement elicitation process and to assure higher confidence in the requirement's quality.

According to [21], academic research on requirements engineering provides valuable insights but it frequently lacks the applicability and granularity needed to tackle specific industry problems. Furthermore, they claim that the absence of a shared framework for comparing theoretical findings complicates the alignment between academic and industrial perspectives. This makes it essential to bridge the gap between theory and practice, ensuring that research outcomes are directly applicable to real-world scenarios. Thus, Frattini [21] explores real-world challenges that organizations face in managing requirements, such as ambiguity, stakeholder involvement, and complexity, highlighting the gap between theory and practice in assessing requirements quality. Furthermore, Frattini et al. [22] contribute to the critical study of quality factors in requirements and their impact on software activities, proposing a harmonized Requirements Quality Theory (RQT), that integrates ideas from both academia and industry, and incorporates advancement to the Quamoco framework [23] with adaptation to requirements engineering. Finally, they propose a roadmap to guide future research and improvements, to support researchers in enhancing requirements quality.

Regarding the effects of smells, we know that requirement smells can have negative effects on software development [4,5]; however, the effects of smells might be null or even positive in some circumstances [10]. For instance, regarding underspecification [7], a sub-smell of the Incomplete smell, we know from the literature that requirements get more specified over time as the clients get a better understanding of what they want [24]. 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 [25]. Conversely, 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.

Regarding how the perceived severity of the same smell varies across project domains, we know that requirement smells can cause rework and time and cost overruns [4,5], and in some cases, a smell might even be catastrophic. For instance, requirements concerning the performance of the system are key for HRT systems [26]. 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.

2.1. Automated requirement smells detection

Requirements Engineering is acknowledged as an expensive, time-consuming, and error-prone process [27,28]. 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. [7], 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. [10] on analysing the syntax of requirements expressed in natural language, developed a tool called *Smella*, which implements part of the Requirements Engineering standards, including ISO/IEC/IEEE 29148:2018 [15], and evaluated its effectiveness in supporting the requirement elicitation process. Similarly, Seki et al. [29] designed a tool to detect 22 "bad smells" (i.e., requirement smell sub-attributes) in use case descriptions written in structured natural language, demonstrating strong performance in terms of precision and recall.

Veizaga et al. [6] developed a tool called *Paska* based on Rimay [30], 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

This subsection discusses empirical evaluations on requirements smells as they provide the foundational context for understanding the significance of our approach and how it extends previous studies through an interview+survey approach.

To the best of our understanding, the closest empirical evaluation to our study is [9]. 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 + survey) 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. [31] 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 [32], in which it emerges that companies with a "high-maturity rating", i.e. companies claiming to follow the best Requirements Engineering practices as a part of their quality management process, experience the same Requirements Engineering problems of companies with lower scores, remarking the necessity of looking deeply inside Requirements Engineering practices.

3. Research methodology

This section describes the methodology we adopted in this research which features an interview and a survey. We started with interviews to gather detailed, qualitative insights that helped us identify key findings and wrong assumptions. These insights guided the design of our survey, enabling us to validate and expand on the findings with a larger group of participants. This approach enhanced the scalability and generalizability of the results while leveraging the larger and more diverse survey sample. Specifically, Fig. 1 describes the methodology we use in terms of activities as steps (arrows) and artefacts as output or inputs of steps (boxes). Our methodology consists of 9 steps and 8 artefacts that we summarize here and elaborate on in the remainder of the paper. We first use an interview (steps 1 to 4) to distil a set of candidate findings to characterize requirements smells in terms of frequency, severity, and effects. Afterwards, we used a survey (steps 5 to 9) to support and dig into the findings by leveraging a larger population.

3.1. Industrial context and population sampling

The context of this research is MBDA Spa, a multinational defence company specializing in the defensive and aerospace domains. The company works closely with organizations to provide advanced defence solutions. Its 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 about 15.000 employees spread across Italy, France, Germany, and the United Kingdom.

Fig. 2 shows the different samples of our Company population (A), i.e. the entire Software Engineering function's population of the MBDA Italy Spa, which is spread over three major sites featuring 122 employees. To produce the Interview sample (B) we employ a disproportionate stratified sampling method [33]. The interviewees were selected in collaboration with the Software Engineering Head of Department at MBDA Italy, who identified candidates based on their proven expertise and experience in the field. Specifically, interviewees were chosen for their involvement in large-scale, long-duration projects and their direct experience with challenges related to requirements issues. Concerning the Pilot-Interview sample (C), we use a random sampling method from the Interview sample (B). Concerning the Pilot-Survey sample (D), we randomly sample one person from the Company Population (A). The survey targeted our entire (company) population.

3.2. Interview

The top part of Fig. 1 describes the methodology we use for the Interview. In Step 1 we analyse literature related to requirements smells and various interview protocols. With this analysis we develop the Interview Protocol 1.0. Afterwards, in Step 2 we conduct the pilot interview using Interview Protocol 1.0 with the Pilot-Interview population. Feedback from this pilot is then used to refine the protocol, resulting in Interview Protocol 2.0. In Step 3 we provide the Interview Protocol 2.0 to the Interview Population and collect their responses, which are referred in Fig. 1 as the Interview Results. Finally, in Step 4 we analyse the Interview Results and identify a set of six Candidate Findings.

Concerning both Interview Protocols 1.0 and 2.0, we use a qualitative semi-structured interview method, which has been proven to be a

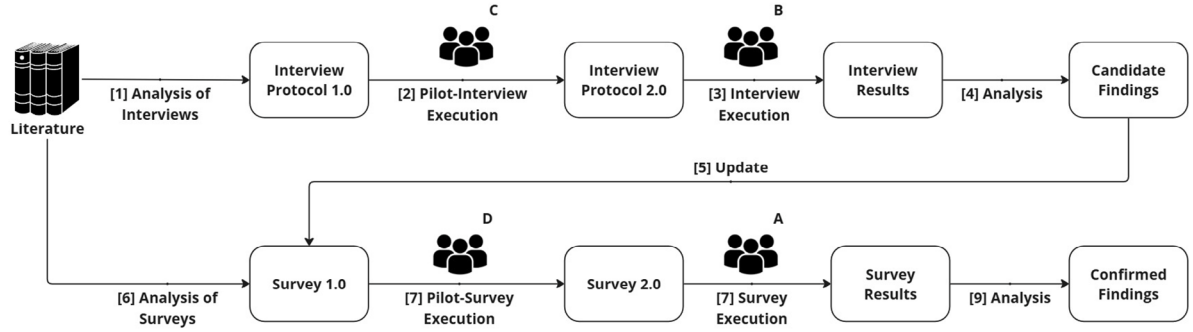


Fig. 1. Overview of our methodology in terms of activities (arrows) and used or produced artefacts (boxes).

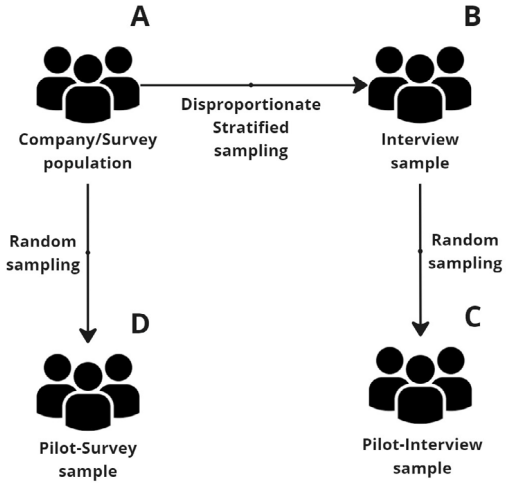


Fig. 2. Sampling strategies for the different populations used in the activities of our methodology.

flexible instrument for investigating areas of interest whose boundaries are unclear or incomplete [34].

The Interview is organized in three sections: **Interviewee characterization**, **Project characterization** and **Requirement Smells**. In the first section, we aim to characterize the participants in terms of role, i.e., the current official position in the company, and years of experience in SW development, while in the second we ask for 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. In the **Requirements Smells** section we focus on interviewees' perception of the severity and frequency of requirements smells, investigating possible effects of smells and contexts that could mitigate or amplify them, and finally asking for possible metrics that could be used for measuring the effects on SW projects. Concerning types of smells, we use the categorization proposed by Montgomery et al. [7]: *Ambiguity*, *Completeness*, *Complexity*, *Consistency*, *Correctness*, *Traceability*, *Reusability*, *Understandability*, *Redundancy*, *Verifiability*, *Relevancy* and *Undefined*. We refer to Montgomery et al. [7] for their definitions.

To investigate which smell is particularly severe or frequent, we use the approach adopted by Fernández et al. [9], asking about the three most and least severe smells, and the three most and least frequent smells.

The complete list of interview questions is reported in Appendix A.

Concerning the fifth question in Requirement Smells section (Appendix A), we aim to maintain alignment with metrics that could be extracted directly from Configuration Management tools. This approach is intended to facilitate the extension of results to generic software development processes. Moreover, due to the limited number of participants, we concentrated on identifying as many metrics as possible rather than establishing direct connections with specific smells, leaving this aspect for the survey stage. We note that the fifth question was not reported in the previous version of this paper [11] due to space constraints.

Table 2
Interviewees and their projects.

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	STX	21	300	7	250K	2	15	SRT
I5	SGL	21	750	7	70K	3	10	SRT
I6	SE	3	200	3	8K	5	1	HRT
I7	HOD/STX	23	400	12	50K	4	20	SRT
I8	SE	18	2000	12	250K	6	18	SRT
I9	SGL	12	100	6	70k	5	1	HRT
I10	SE	16	2000	12	250k	6	18	SRT

Table 2 reports data about the population and project characterization of the interview. As reported in Table 2, a good mix of roles has been achieved (4 SW Group Leaders, 3 SW Engineer, 2 SW Technical Experts and 1 SW Project Leader), with an average work experience in requirements analysis of 16.8 years. Similarly, projects' characteristics demonstrated significant variety: in fact, software development teams range in size from 4 to 12 people, while the number of requirements per project varies between 200 and 2000. The number of Lines of code also exhibits a significant spread, ranging from 8K to 300K, encompassing both Soft Real-Time and Hard Real-Time domains.

3.3. Survey

Our survey aims to support or refute the candidate findings about requirement smells, that we identified through the interview.

The lower part of Fig. 1 describes the methodology we use for the survey design. In Step 5, we update the Candidate Findings for standardization purposes as we adopt the INCOSE categorization for requirements smells definition [17]. In Step 6 we analyse relevant survey literature to develop the first iteration of the survey, i.e. Survey 1.0. In Step 7, we conduct a pilot survey and use the feedback to refine the survey, resulting in Survey 2.0. In Step 8 we conduct Survey 2.0 with the Survey population, and in Step 9 we analyse the survey results to support and eventually expand upon the identified findings.

We now describe Survey 2.0 which is organized into 4 sections: **Introduction** , **Demographics**, **Agreement with Findings** and **Smells-to-Effects linking**.

In the **Introduction** section, we explain the reasoning behind the survey and ask for explicit consent to use participants' data for research purposes. We emphasize that all results will be anonymized, reported

only in aggregate form and that providing feedback and opinions will have no repercussions on participants' jobs. As suggested in [35], this reminder is important because the commitment to confidentiality and ethical data practices is essential to establishing trust with participants and ensuring the integrity of our research findings.

In the **Demographics** section, we ask the participants about their roles in the company and their experience in requirements analysis, using both Single Choice (SC) and Multiple Choice (MC) types of questions. To ensure clarity and a common understanding of the presented types of requirements, we included concise definitions alongside each one, readily accessible for reference throughout the survey.

In the **Agreement with Findings** section, we investigate the output of the interview [11], asking the participants their level of agreement on the candidate Findings (F). Afterwards, in the **Smells-To-Effects Linking** section, we expand the output of the interview, presenting a list of metrics and asking the participants to indicate the extent to which each metric is suitable for measuring the impact of a specific smell. The complete list of survey questions is reported in Appendix B.

As a set of smells, we slightly updated the smells from the interview according to INCOSE standards [17]. Specifically, the INCOSE standard provides the following desired characteristics of requirements: *Necessary, Appropriate, Unambiguous, Complete, Singular, Feasible, Verifiable, Correct, Consistent and Conforming.* We updated the smell categorization from the one used in the interview for two reasons:

1. The term smell has a negative connotation in the one used in the interview whereas the INCOSE standard provides the desired characteristics of requirements. Therefore, we defined, and used, the following set of smells as the one-to-one absence of the above-mentioned desired characteristics: *Unnecessary, Inappropriate, Ambiguous, Incomplete, Plural (Non-singular), Infeasible, Unverifiable, Incorrect, Inconsistent, Non-conforming*.
2. The adherence to the INCOSE standard, since a well-established international standard, enabled us to exploit our research results in many departments of the company and across countries.

Table 1 reports the smell categorization and their mapping, which we use in the interview as derived from Montgomery et al. [7] and in the survey, as derived from the INCOSE standard [17]. We mapped the two categorization by analysing the definitions provided by INCOSE and the ones in Montgomery et al. [7]. According to Table 1, the undefined smell in Montgomery et al. [7] has no related smell in the INCOSE categorization. Moreover, some smells of the Montgomery et al. [7] insist on more than one smell in the INCOSE categorization. We note that the subjects were exposed to one categorization at a time. Finally, our results are related to smells in general and not specific smells.

The Likert scale is a "psychometric scale that has multiple categories from which respondents choose to indicate their opinions, attitudes, or feelings about a particular issue" [36]. Among the different types of Likert scales, we use a Likert scale with a neutral point for the following reasons [37]:

- Balanced Response Options: including a neutral point in a likert scale supports balance in the response options, allowing respondents to express neutrality when they neither agree nor disagree strongly with the statement.
- Avoiding Response Bias: without a neutral point, respondents might feel compelled to choose between agreement and disagreement even if they feel neutral about the statement. This can introduce response bias and distort the data.

The 100-point prioritization is a comparative scaling technique where respondents are tasked with distributing 100 points among a set of options to indicate relative importance or preference [37], and, in the Software Engineering field, it is largely used as a requirement prioritization technique [38]. We decide to apply the 100-point prioritization for two main reasons:

Table 3
Survey techniques adopted for Findings investigation.

Findings	Investigation technique	Analysis
F1	100P	Distribution, RBLMM Test, WSR Test
F2	LKS	Frequency
F3	100P	Distribution, RBLMM Test, WSR Test
F4	LKS	Frequency
F5	LKS	Frequency
F6	LKS	Distribution, Bubble, MCA

- support that different smells have different severity or frequency, and
- identify which smell is more/less severe or frequent than the others.

The list of findings to support or challenge in the survey is hereafter reported:

- **F1**: The perceived severity varies across types of smells
- **F2**: The perceived severity of the same smell varies across project domains
- **F3**: The perceived frequency varies across types of smells
- **F4**: The frequency of a smell is perceived differently across roles and phases
- **F5**: The severity of a smell might change across roles or phases
- **F6**: The perceived effects may vary across types of smells

In this research context, we were particularly interested in investigating the effects of smells and their differences. Therefore, to better investigate the effects of smells, we asked participants, in addition to the agreement to F6, the specific process and product metrics we could use to measure the impact of a specific smell, via a multiple choice (MC) type of question. From the interview, we identified the following set of 12 candidate metrics:

- **M1**: High number of Defects - i.e. bugs during the software development phase
- **M2**: High number of Queries - i.e. requests of clarification about requirements content
- **M3**: High number of Change Requests of the TRS (Technical Requirement Specification) document
- **M4**: High number of code Commits
- **M5**: High number of LOC (lines of code) tested by inspection
- **M6**: High number of Line of Code impacted by Safety Critical standards
- **M7**: Low score for code Quality Assurance - e.g. the presence of duplicate code, high cyclomatic complexity, deep nesting, long methods, inappropriate comments, etc...
- **M8**: High number of Technical Facts - i.e software bugs in production, coming after a real deployment of a system on the field
- **M9**: High number of document updates - i.e. any documents produced during the SW development lifecycle
- **M10**: High number of test updates/reworks
- **M11**: Delay in SRS (Software Requirement Specification) document definition
- **M12**: Delay in starting coding activities
- **M13**: Others — to let the participants specify other metrics or better specify their choices

Regarding data analysis, Table 3 reports the specific analyses performed to investigate specific findings. Similarly, we analysed 100P by using box and whisker plot to visualize the point assigned for each smell across different participants. To investigate whether the severity points, as provided by survey respondents, differ across smells, we use a rank-based linear mixed model (RBLMM). This method is ideal for our repeated-measures design, where the same participants provided scores for multiple smells, resulting in independent data. By ranking the frequency ratings before analysis, we ensure that the test was robust to deviations from normality, making it a nonparametric alternative suited to our dataset. The model examines Smell Type as a fixed effect to detect differences in severity scores and accounts for individual variability by including survey respondents' ID as a random effect. This approach is aligned with recommendations for analysing ranked data in mixed designs, as outlined by [39], and allowed us to test for differences across smells while controlling for within-subject correlations and avoiding strict assumptions about the data distribution. We apply the same approach to both severity and frequency.

To better understand the significant differences revealed by the rank-based linear mixed model, we conduct a pairwise comparison between smell categories using the Wilcoxon Signed-Rank (WSR) test. This nonparametric test is well-suited for repeated-measures designs because it accounts for the dependency between observations, ensuring that paired data are appropriately analysed. Unlike parametric methods, the Wilcoxon Signed-Rank Test does not assume normality, making it a robust tool for evaluating median differences between smell categories. This approach enabled us to pinpoint specific smell categories that showed significant differences in severity scores while preserving the rigour of the statistical analysis [40]. We apply the same approach to both severity and frequency. Given the numerous statistical tests, we set *alpha* to 0.01 to avoid type 1 error compared to a standard 0.05 used in software engineering [41].

Finally, we analyse correspondence between metrics and smells by using bubble plots and Multiple Correspondence Analysis (MCA). MCA is a statistical technique generally used to analyse and represent the relationships and patterns existing among multiple categorical variables, in a two-dimensional space [42]. While not commonly utilized in software engineering, MCA proves especially beneficial when dealing with a limited number of variables (three or fewer) and when those variables have numerous categories, as it enables a comprehensive overview of the data for easier interpretation. MCA generates a two-dimensional plot where each point corresponds to a specific category of a categorical variable, and the distance between points reflects the strength of association between the respective categories. The first dimension represents the primary axis capturing the largest variance (or inertia) in the data, highlighting the most significant patterns or contrasts, which, in our case, are the dominant relationships between specific metrics and smells. The second dimension, orthogonal to the first, accounts for the second-largest variance, often reflecting secondary relationships or subtler distinctions. The percentages associated with each dimension indicate their contribution to explaining the overall variation, while the Lambda values quantify the inertia captured by each variation [42]. Thus, when categories from different variables appear close together on the plot, it indicates a strong association between them. Furthermore, if all categories of the same variable cluster closely, it suggests minimal differences among them, indicating a lower overall impact of that variable. In this study, we satisfy the criteria for employing MCA effectively [42], as we are examining only two discrete variables, i.e. Metric and Smell, with multiple levels.

As suggested in [43], in Step 7, we validate the survey by submitting it to a small set of respondents (referred to as "Pilot-Survey population (D)" in Fig. 2) and collecting their suggestions to ensure the highest quality. We report that no substantial modifications have been made to the survey after the test phase.

Regarding the survey execution (Step 8) we monitor the percentage of respondents over the designated three-week execution period, and after two weeks we issued a reminder to the entire population, inviting those who had not started or completed the survey to proceed with completion and submission.

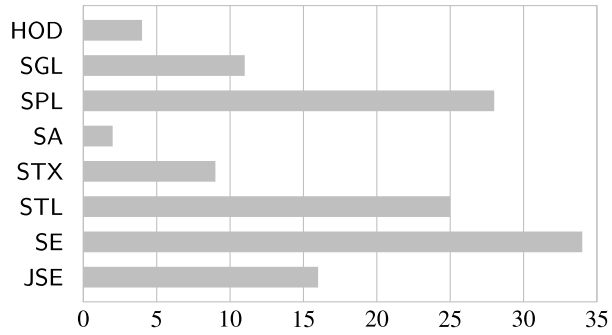


Fig. 3. Distribution of roles within the SW Engineering function.

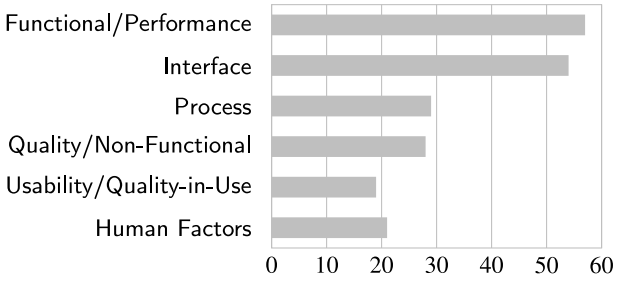


Fig. 4. Distribution of the types of requirements the participants have ever been involved with.

4. Results

In this section, we present the results of the interview and the survey. After analysing the demographics, we report our results in the form of six "Findings".

4.1. Demographics

If on the one side, all targeted ten interviewees answered our questions, only some of the surveyees answered. Specifically, out of the 122 targeted surveyees, only 73 answered (60% of acceptance rate). Out of these 73 answers, we discarded 14 because were incomplete and 1 because declining consent for data usage, ending up with a sample of 58 participants, that is, a 48% participation rate.

Fig. 3 reports the distribution of roles inside MBDA Italy S.p.a. Software Engineering function. According to Fig. 3, there is a wide distribution of roles, with the majority of positions being occupied by SW Engineers (34). The limited presence of certain roles like SW Architects (2) and SW Head of Departments (4) is expected and reflects the company's hierarchical structure.

Fig. 4 reports the distribution of types of requirements that the participants have been worked with. According to Fig. 4, the most frequent types of requirements are Functional/Performance (98%) and Interface (93%) while the less frequent are Usability/Quality-in-Use (32%) and Human Factors (36%) requirements. It also emerged that 29 participants (50%) have experience in the domain of Hard Real-Time software. The distribution of types of requirements reflects a holistic approach to software development, reasonably stemming from the company's core business. The prevalence of the Functional/Performance and Interface types of requirements indicates a pronounced focus on developing software that not only meets functional specifications but is also highly standardized and easily integrable with other software components. Additionally, the significant presence of Process and Quality/Non-Functional requirements suggests that practitioners are expected to manage aspects such as software reliability and maintainability and to innovate and refine the company's methodologies, workflows, and procedures.

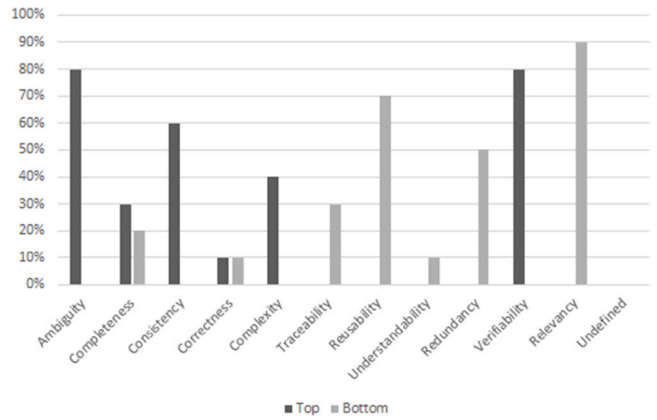


Fig. 5. Interview — Distribution of the three most and least severe requirement smells.

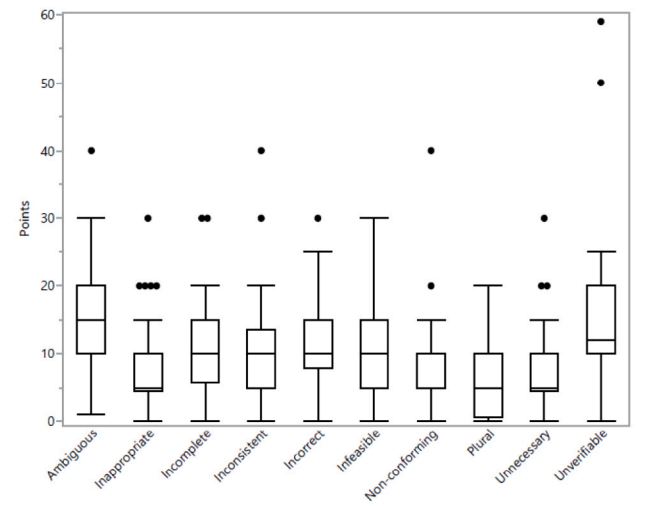


Fig. 6. Survey — Requirements smells severity distribution as perceived by practitioners.

4.2. F1: The perceived severity varies across types of smells.

Interview. Fig. 5 reports the frequency distribution of the three most and least severe requirement smells. According to Fig. 5, the perceived most severe smells are *Ambiguous* (80%), *Unverifiable* (80%), and *Inconsistent* (60%). The perceived least severe smells are *Non-Relevant* (90%), *Non-Reusable* (70%) and *Redundant* (50%). Interestingly, only *Incomplete* and *Incorrect* smells have been identified as the most and least three severe by different interviewees.

Survey. Fig. 6 reports the requirements smells severity distribution as perceived by practitioners. According to Fig. 6, the smells that are perceived as the most severe are *Ambiguous*, *Unverifiable*, *Incorrect*, and *Infeasible*, while *Plural*, *Non-conforming* and *Unnecessary* requirements are perceived as less severe.

The resulting *p-value* from the rank-based linear mixed model testing the difference in severity among smells is less than alpha, and therefore, we can claim that there is a difference in severity across smells. Regarding the comparison between pairs of smells in terms of severity, Table 4 reports the resulting p-values of the Wilcoxon Signed-Rank Test. According to Table 4, 31 out of 45 combinations of smells resulted as statistically different.

4.3. F2: The perceived severity of the same smell varies across project domains

Interview.

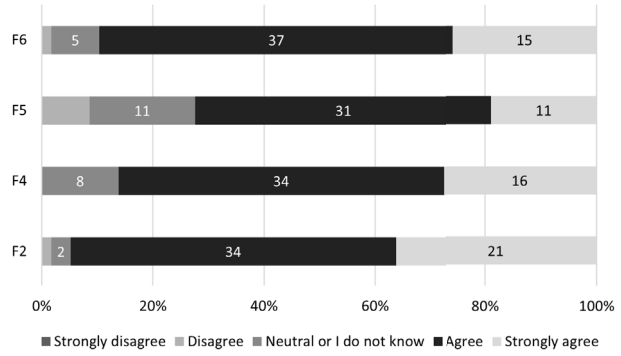


Fig. 7. Survey — Practitioners agreement to Findings.

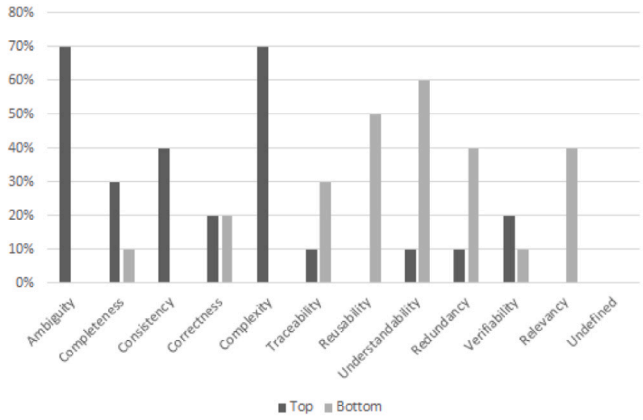


Fig. 8. Interview — Distribution of the top three and bottom three requirement smell frequency.

Our interview reports that "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)

Survey. Fig. 7 reports the agreement distribution of the participants to findings. According to Fig. 7, participants expressed a strong agreement with F2. Specifically, 36% strongly agreed, 59% agreed, 4% neutral, 1% disagreed, and none strongly disagreed with F2.

4.4. F3: The perceived frequency varies across types of smells.

Interview. Fig. 8 reports the frequency distribution of the most and least frequent requirement smells. According to Fig. 8, the perceived most frequent smells are *Ambiguous* (70%), *Complex* (70%) and *Inconsistent* (40%). The perceived least frequent smells are *Not-understandable* (60%), *Not-Reusable* (40%), and *Not-Relevant* (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.

Survey. Fig. 9 reports the requirements smells frequency distribution as perceived by the practitioners. According to Fig. 9, the perceived most frequent smells are *Ambiguous*, *Incomplete*, *Incorrect* and *Unverifiable*, while the least frequent are *Infeasible*, *Non-Conforming*, *Unnecessary* and *Inappropriate*.

The resulting *p-value* from the rank-based linear mixed model testing the difference in frequency among smells is less than alpha, and

Table 4
Survey — Wilcoxon Signed-Rank test results (p-value) for each combination of smells,in terms of severity.

	Ambiguous	Inappropriate	Incomplete	Inconsistent	Incorrect	Infeasible	Non-conforming	Plural	Unnecessary	Unverifiable
Ambiguous	–	*0,0001*	*0,0001*	*0,0001*	*0,0003*	*0,0001*	*0,0001*	*0,0001*	*0,0001*	0,0915
Inappropriate		–	*0,0062*	*0,0065*	*0,0004*	0,0181	0,4545	0,1185	0,6373	*0,0001*
Incomplete			–	0,8628	0,3094	0,8795	*0,0001*	*0,0001*	*0,0008*	*0,0056*
Inconsistent				–	0,2048	0,9955	*0,0001*	*0,0001*	*0,0008*	*0,0023*
Incorrect					–	0,2821	*0,0001*	*0,0001*	*0,0001*	0,0648
Infeasible						–	*0,0009*	*0,0001*	*0,004*	*0,0055*
Non-conforming							–	0,4121	0,8574	*0,0001*
Plural								–	0,2977	*0,0001*
Unnecessary									–	*0,0001*
Unverifiable										–

Table 5
Survey — Wilcoxon Signed-Rank test p-values for each combination of smells in terms of frequency.

	Ambiguous	Inappropriate	Incomplete	Inconsistent	Incorrect	Infeasible	Non-conforming	Plural	Unnecessary	Unverifiable
Ambiguous	–	*0,0001*	*0,0022*	*0,0001*	*0,0001*	*0,0001*	*0,0001*	*0,0001*	*0,0001*	*0,0001*
Inappropriate		–	*0,0001*	0,0255	*0,0001*	0,0682	0,1342	0,1862	0,9057	*0,0003*
Incomplete			–	*0,0002*	0,0212	*0,0001*	*0,0001*	*0,0001*	*0,0001*	0,0100
Inconsistent				–	0,0973	*0,0001*	*0,0004*	0,4283	0,0214	0,1390
Incorrect					–	*0,0001*	*0,0001*	0,0105	*0,0001*	0,7031
Infeasible						–	0,868	*0,0017*	0,0402	*0,0001*
Non-conforming							–	*0,0063*	0,0859	*0,0001*
Plural								–	0,2003	0,0292
Unnecessary									–	*0,0002*
Unverifiable										–

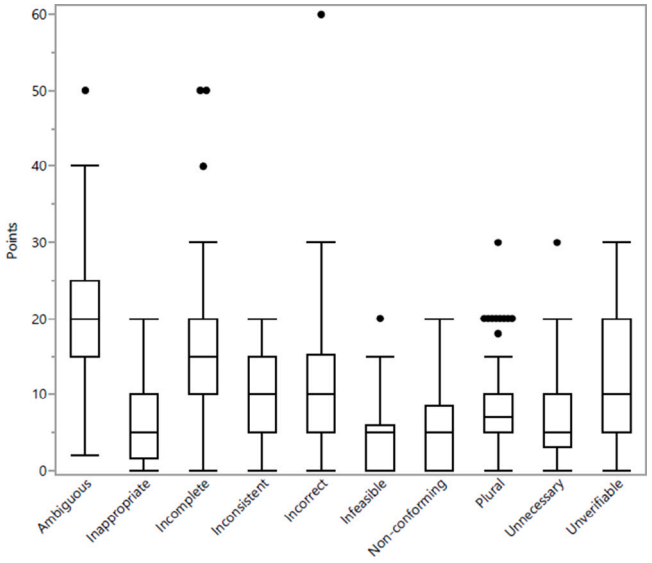


Fig. 9. Survey — Requirements smells frequency distribution as perceived by practitioners.

therefore, we can claim that there is a difference in frequency across smells. Regarding the comparison between pairs of smells in terms of frequency, Table 5 reports the resulting p-values of the Wilcoxon Signed-Rank Test. According to Table 5, 34 out of 45 combinations of smells resulted as statistically different.

4.5. F4: The frequency of a smell is perceived differently across roles or phases.

Interview. 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)

Survey. According to Fig. 7, participants expressed a strong agreement with F4. Specifically, 28% strongly agreed, 59% agreed, 13% neutral, and none disagreed nor strongly disagreed with F4.

4.6. F5: The severity of a smell might change across roles or phases.

Interview. If, 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.

Survey. According to Fig. 7, participants expressed an overall agreement with F5. Specifically, 19% strongly agreed, 53% agreed, 19% neutral, 9% disagreed, and none strongly disagreed with F5.

Fig. 7 suggests a mixed response to the presented F5, realizing the lowest agreement among all the Findings: 53.4% of participants agreed with the statement, and only 19.0% strongly agreed. On the opposing side, 8.6% disagreed with the assertion. However, a noteworthy 19.0% remained neutral or uncertain about the described effects. F6 results also the Finding with more comments (2 from the Disagree group, and 3 from the Agree group). A detailed analysis of the comments will be reported in the Discussion section.

4.7. F6: The perceived effects may vary across types of smells.

Interview. Intuitively, specific smells cause specific problems; however, the specific effects of specific requirement smells, to the best of our knowledge, are unknown. We now discuss the examples of the *Unverifiable*, *Ambiguous*, and *Complex* smells.

Regarding the *Unverifiable* smell, 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 *Unverifiable* 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 *Ambiguous* smell, if a requirement is unclear, this will likely need to go back and forth between requirements engineers and developers to identify and formalize 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)

Finally, regarding *Complex* smell, 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 *Unverifiable* and *Ambiguous* smells do not reasonably lead to complex code. Similarly, *Complex* smell does not reasonably lead to decreased customer satisfaction. Thus, the impact of requirement smells is perceived as varying across smells.

Survey. According to Fig. 7, participants expressed a strong agreement with F6. Specifically, 26% strongly agreed, 64% agreed, 9% neutral, 1% disagreed, and none strongly disagreed with F6.

Fig. 10 reports frequencies of metrics that are perceived as useful for measuring the effects of specific requirements smells. According to Fig. 10, some metrics are particularly frequent at detecting specific smells. For instance, metrics M1 (High number of Defects), M2 (High number of Queries), M3 (High number of Change Requests of the TRS document), M9 (High number of document updates), and M10 (High number of test updates/reworks) are perceived as useful for measuring the effect of *Ambiguous*, while M5 (High number of LOC tested by inspection) and M10 (High number of test updates/reworks) are perceived as useful for measuring *Unverifiable* smell.

Fig. 11 reports the multiple correspondence analysis between requirements smells and metrics for measuring their effects. According to Fig. 11, some metrics are particularly close to specific smells: for example M5 (High number of LOC tested by inspection) and M6 (High number of Line of Code impacted by Safety Critical standards) with *Unverifiable*, M3 (High number of Change Requests of the TRS document) with *Incomplete*, M4 (High number of code Commits) with *Incorrect* and M8 (High number of Technical Facts) with *Non-Conforming*, and finally M11 (Delay in SRS document definition) with *Plural*.

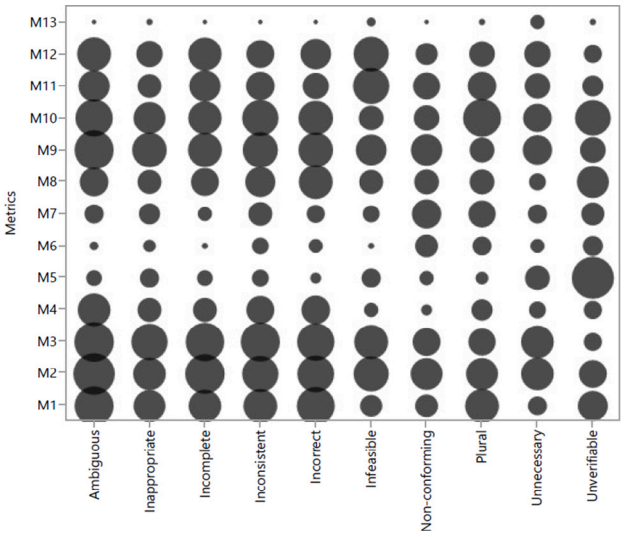


Fig. 10. Survey — Frequencies of metrics that are perceived as useful for measuring the effects of specific requirements smells.

5. Discussion

Our investigation into requirements smells provides valuable insights into how practitioners perceive these quality concerns, informing both software engineering practitioners and researchers. We now discuss each of the six Findings by analysing the agreement between interview and survey data, variations within the findings, their connection to the reviewed literature, and their implications for practitioners and researchers. To facilitate the understanding of the results, we now refer to the smells by using the Survey categorization only; see Table 1 for the interview categorization mapping.

5.1. F1: The perceived severity varies across types of smells.

Both interviews and surveys identified *Ambiguous* and *Unverifiable* as the most severe types of requirement smell. Considering Table 1, the severity perception of *Infeasible* requirements from the survey aligns with that of *Ambiguous* requirements from the interviews. In contrast, the *Inconsistent* smell, which was regarded as significantly severe by interviewees, was perceived as only moderately severe by survey participants. Meanwhile, the *Incorrect* smell was identified as severe by survey participants and just one interviewee. Similarly, survey participants perceived *Plural*, *Non-conforming*, and *Unnecessary* as the least severe requirement smells, aligning with the interview findings. According to the categorization mapping presented in Table 1, the *Relevant* and *Redundant* smells from the interview categorization map to the *Unnecessary* smell, while the *Non-reusable* smell maps to the *Plural* smell, further confirming the interview results.

Our survey results confirm our interview Finding 1 since our statistical analysis allows us to reject the hypothesis that the perceived severity is equal across smells. Regarding the comparison between specific pairs of smells, in terms of severity, according to Table 4, we observe that *Ambiguous* shows a significant difference to all other smells other than *Unverifiable*. This result could support the conjecture that there is no difference in severity between such two smells. However, although this result seems intuitive, the survey design does not allow supporting it since the absence of evidence does not imply evidence of absence [44]. Since, according to Fig. 6, *Ambiguous* is the smell with the highest severity, and *Unverifiable* comes second, we can assert that these two smells are the two most severe smells, and all the others are significantly less severe.

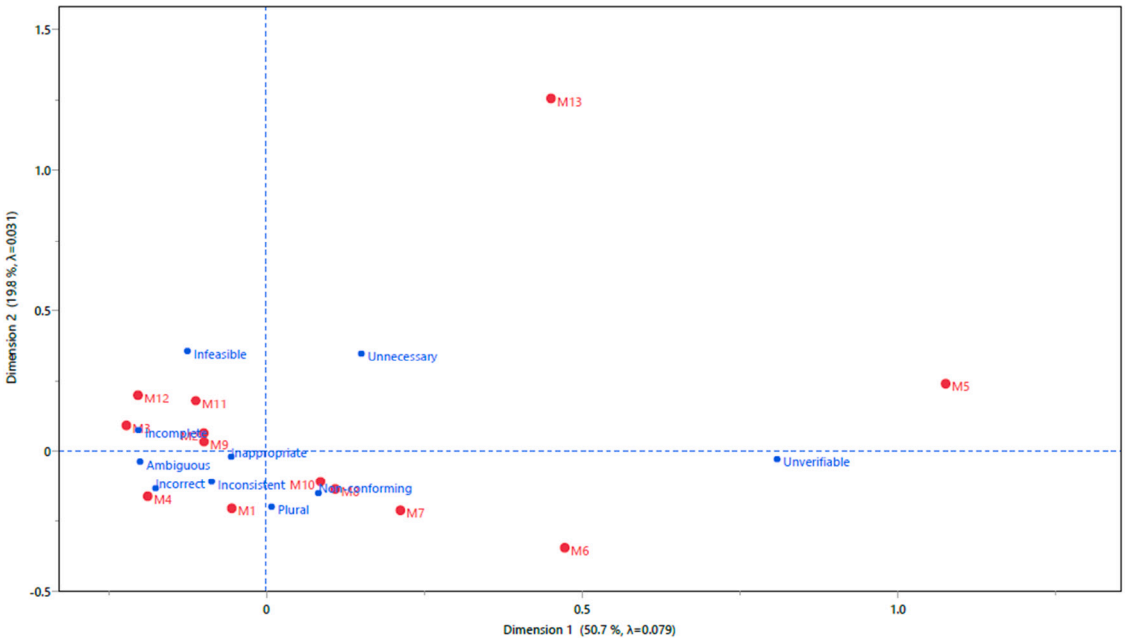


Fig. 11. Survey — Multiple correspondence analysis between requirements smells and metrics for measuring their effects.

The emphasis on *Ambiguous* and *Incorrect* aligns with prior research by Montgomery et al. [7], who identified these as the most studied quality attributes in requirements engineering. Conversely, Montgomery et al. [7] identified Verifiability as the least studied, whereas *Unverifiabile* smell result as the most severe.

These results suggest that practitioners should employ early detection and mitigation strategies for these high-severity smells during development. Since Verifiability appears among the least studied requirements quality attributes [7], researchers are invited to further investigate the causes and effects of this smell and its connection with *Ambiguous* smell.

5.2. F2: The perceived severity of the same smell varies across project domains

In HRT contexts, the *Unverifiable* smell results particularly severe due to the high stakes of system performance and safety [26]. However, in less critical domains, such as non-safety-critical business applications or e-commerce systems, the impact of unverifiable requirements may be less pronounced. In such cases, this smell might only result in delays or inefficiencies rather than posing significant risks to system functionality or user safety [4,5]. From this perspective, a high level of consensus (95%) emerged among survey participants regarding the interview finding: the perceived severity of smells varies across project domains.

This result aligns closely with [12] research, emphasizing the criticality of domain-specific considerations in requirements engineering. Sommerville and Sawyer [12] highlight the importance of clear and concise requirements, particularly in safety–critical domains, where ambiguous or incomplete requirements can lead to catastrophic consequences. This notion is supported by Intrigila et al. [45], who highlight that managing requirements effectively in critical systems – such as safety–critical or mission-critical projects – is essential to maintaining dependability and avoiding costly regressions during software evolution. Additionally, research from Firesmith [46] emphasizes the categorization of safety-related requirements to manage risks associated with critical systems, stressing that varying severity levels (e.g., catastrophic versus minor) must be contextually assessed to ensure that system requirements mitigate risks appropriately across domains. For instance, in safety–critical systems, even minor deviations in requirement quality can escalate into severe operational risks. Our findings reinforce these notions by suggesting that smells can have a more significant impact depending on the project's domain. Researchers could investigate methods to prioritize smell detection and mitigation specific to targeted project domains. By integrating tailored mitigation strategies, such as domain-specific requirement refinement, practitioners can address these smells more effectively and prioritize issues based on their domain-specific impact. Practitioners shall be aware of the amplified risk associated with requirement smells in critical projects.

5.3. F3: The perceived frequency varies across types of smells.

The survey supports that the perceived frequency varies across types of smells. Specifically, both interview and survey suggest that *Ambiguous*, *Incomplete*, *Unverifiable* and *Incorrect* are the smells that are perceived as most frequent. We note that *Ambiguous*, *Unverifiable* smells resulted in being perceived as both the most severe (F1) and the most frequent.

Our survey results confirm our interview Finding 3 since our statistical analysis allows us to reject the hypothesis that the perceived frequency is equal across smells. Regarding the comparison between specific pairs of smallest, in terms of frequency, according to Table 4, we observe that *Ambiguous* shows a significant difference from all other smells. Since, according to Fig. 9, *Ambiguous* is the smell with the highest frequency, and *Incomplete* comes second, we can assert that these two smells are the two most frequent smells, and all the others are significantly less severe.

This result aligns with the focus on ambiguity, incompleteness, and inconsistency highlighted in Montgomery et al. [7] and Seki et al. [29], and it suggests the need for improved communication and requirement elicitation techniques during the early stages of software development, as remarked in [47]. Further supporting this, the study by Muhamad et al. [48] remarks that ambiguity in software requirements is a significant factor in software project failures. Their work describes ambiguity as stemming from natural language's inherent imprecision, which can lead to inconsistent interpretations and costly rework. They advocate for leveraging machine learning techniques to detect and mitigate ambiguity in requirements documents, which resonates with the need for systematic smell detection highlighted in the survey. Lastly, Berry and

Kamsties [49] provide a comprehensive taxonomy of ambiguity types in requirements, such as lexical and syntactic ambiguities, and suggest structured validation techniques to address these issues. Their approach complements the survey's implication that ambiguity detection tools tailored to specific smells could enhance the software development process. We further note that the *Infeasible* smell is perceived as lowly frequent but highly severe, and this result could be because this smell is so important in the safety–critical domain, which is the context of this study, that infeasible requirements have been filtered or corrected during requirement specification.

5.4. F4: The frequency of a smell is perceived differently across roles or phases.

A high level of consensus (87%) emerged among survey participants regarding the interview finding: the frequency of a smell is perceived differently across roles or phases. One possible reason for this result is that as the requirements are managed and improved over the development life cycles, different roles are exposed to different versions, also in terms of quality, of the same requirements. Specifically, one survey participant, playing the role of Software Project Leader, supported this result by explicitly reporting in the comment section that the introduction of a Software Requirement Specification expert in the development process led to fewer smells in the design phase as this new role had as duty also the identification and removal of smells.

The finding that the frequency of smells is perceived differently across roles or phases aligns with previous research on role-specific and phase-specific variations in software quality perceptions. For instance, Achimugu et al. [50] highlight that the responsibilities and focus areas of stakeholders, such as developers, testers, and project managers, influence their prioritization and identification of software quality issues. This supports the notion that roles exposed to distinct development stages may encounter different types and frequencies of smells, as identified in this study. Furthermore, Biolchini et al. [51] remark the importance of structured methodologies and role-specific responsibilities, such as integrating experts for requirement specification, to improve overall quality and reduce potential defects, aligning with the finding that introducing specialized roles mitigates smells during design. Additionally, prior work by Mäntylä and Lassenius [52] discusses how perceptions of code and design smells vary between development and maintenance phases, emphasizing the temporal nature of smell identification. This resonates with the finding that requirements management and improvements across development life cycles contribute to differing perceptions of smell frequency. These insights collectively suggest that integrating role-specific responsibilities and phase-aware smell detection strategies could be critical in positively managing and mitigating smell effects. As the requirements are managed over the development life cycles and get improved over the life cycle, researchers could explore how to integrate smell detection tools that are specific to different development stages or various roles.

5.5. F5: The severity of a smell might change across roles or phases.

A high level of consensus (72%) emerged among survey participants regarding the interview finding: the severity of a smell might change across roles or phases. We note that this is the least agreed finding; a noteworthy 19.0% remained neutral or uncertain about this finding. Specifically, one survey participant, playing the role of Software Project Leader, stated that the *Incomplete* smell can have the most disruptive effect on software architecture since completing those incomplete requirements can cause costly changes to the organization of the code. The participant revealed that one possible solution to deal with this smell is the adoption of an architecture that is as open and robust as possible so that it is able to mitigate by construction the impacts of certain smells, especially when coming from changes in non-functional or performance requirements. Another survey participant, playing the role of Software Group Leader, reported that a solution to deal with the *Incomplete* smell is the use of experienced software architects who can better deal with what is missing the requirements. However, this solution does not guarantee the mitigation of the effects of the *Incomplete* smell.

The observation that the severity of a smell may change across roles or phases aligns with and extends findings in prior research. For example, Bavota et al. [53] emphasize the evolving perception of code smells during different phases of software maintenance, where their criticality can escalate depending on architectural decisions and stakeholder priorities, echoing the survey participant's view on *Incomplete* smell. Moreover, previous work by Yamashita and Moonen [54] discusses how developers' roles influence their prioritization of smells. Their study found that project leaders often prioritize smells with broader architectural implications, such as those affecting system maintainability and scalability, rather than local code issues. This directly supports the findings from the current study, where leaders suggested robust architectural frameworks and experienced architects as key solutions to mitigating the impact of incomplete requirements. These parallels suggest that while the awareness of smell impacts varies, the broader theme of adapting strategies to mitigate smells across development phases is widely recognized in the literature. The integration of these findings into practice could guide researchers and practitioners in tailoring smell mitigation strategies that account for role-specific and phase-specific challenges.

5.6. F6: The perceived effects may vary across types of smells.

Regarding the reasons for the interviewees and the survey participants to determine what effects are caused by which smell, we note that the effects of requirement smells were identified through interviews with experienced practitioners, who provided real-world examples of how specific smells impacted their projects. These findings were then validated through the survey, where participants assessed the proposed links between smell types and their effects based on their own practical experience. A high level of consensus (90%) emerged among survey participants regarding the interview finding: the perceived effects may vary across types of smells. The rationale of this finding is that specific smells reasonably cause specific problems. According to Figs. 10 and 11 we can derive the following take-aways:

- Metrics M1(High number of defects), M4 (High number of code commits), M9 (High number of document updates), and M10 (High number of test updates/reworks) are closely clustered, suggesting they are similarly associated with the smells near them.
- The cluster near the centre of the plot indicates metrics and smells with a more neutral or balanced association. This cluster includes smells like *Incorrect*, *Inconsistent*, *Ambiguous*, and *Incomplete* and metrics M1 (High number of defects), M4 (High number of code commits), M9 (High number of document updates), and M10 (High number of test updates/reworks).
- The distinct clusters or outliers, such as *Unnecessary* and *Unverifiable* smells, which are far from the centre, suggest a strong association with specific metrics. For instance, *Unnecessary* seems to be uniquely associated with M13 (Others — participant-specified metrics), while *Unverifiable* is associated with some metrics not closely clustered with others.
- Metrics such as M1 (High number of defects), M4 (High number of code commits), M9 (High number of document updates), and M10 (High number of test updates/reworks) have a similar association with central smells like *Incorrect*, *Inconsistent*, *Ambiguous*, and *Incomplete*.
- The *Unnecessary* smell is strongly associated with M13 (Others — participant-specified metrics), as it is placed far from other smells and metrics. This could mean that participants often specified additional metrics that uniquely relate to unnecessary elements in the project.

- The smell *Unverifiable* is distinctly associated with M5 (High number of lines of code tested by inspection), indicating a strong relationship between high inspection activity and unverifiable requirements.
- Metrics M5 (High number of lines of code tested by inspection) and M6 (High number of lines of code impacted by Safety Critical standards) are placed relatively far from the central cluster, indicating they have unique associations with certain smells. M5 (High number of lines of code tested by inspection) appears to have a strong association with *Unverifiable* smells, and M6 (High number of lines of code impacted by Safety Critical standards) has a moderate association with smells such as *Non-conforming*.

To the best of our knowledge, no prior research has established a direct connection between requirements smells and their measurable effects using metrics derived from configuration control tools. Existing studies such as [41,55], and [10] focus on respectively identifying, categorizing, and automating the detection of requirements smells, particularly at early development stages, to address potential issues arising from poorly written requirements. However, these works stop short of drawing explicit correlations between specific smells and their practical effects on software development processes. As a result, they do not enable prioritization of smells based on their tangible impacts on project metrics. Our study builds upon and extends this foundation by investigating the consequences of requirement smells, enabling both the detection of smells based on their downstream effects and the prioritization of mitigation strategies to address their impact on active projects.

For researchers, the correlation between poorly written requirements and the presence of bugs highlights a significant risk of project failure. For practitioners, the focus shifts towards the practical application of these metrics; i.e., to minimize a specific consequence, such as bugs, practitioners must concentrate on the corresponding smell to identify and address these issues as soon as possible.

6. Threats to validity

Threats to validity are factors that can compromise the credibility or generalizability of the study's results, such as biases in data collection or limitations in the experimental design. As explained in Wohlin et al. [56], addressing these threats transparently helps readers evaluate the trustworthiness of the findings and their applicability to real-world scenarios. This section reports threats to validity identified in our study, and the way we mitigated them. The section is organized by threat type, i.e., Internal, Construct, External, Conclusion [56] and Reliability [57].

6.1. Internal validity

Internal validity concerns the influences that can affect the independent variables concerning causality [56]. A possible threat to internal validity is the potential selection bias in both interviews and surveys. In the interviews, despite the fact that we selected participants with a representative proportion of roles, there is still a risk that the subjects may have been biased towards specific answers. Although participants were selected to achieve a proportionate mix of roles, the possibility of selection bias remains. To mitigate this, we achieved a 100% acceptance rate, indicating willingness across all invited participants. Similarly, the survey's 60% acceptance rate poses minimal threat to validity given the achieved demographic diversity. As shown in Fig. 3, the sample includes a balanced distribution of roles across various departments within the Software Engineering function, suggesting it is representative of the practitioner population. When inquiring about perceived severity, we avoided specifying levels to minimize subjective interpretations by interviewees and participants.

We note that the years of experience in software development, combined with role-specific responsibilities, provide a strong indication of expertise in software requirements and RE smells. There is a risk that interviewees and survey participants did not have enough experience to provide valuable answers. To mitigate this threat to validity during the selection process, the participants in our study were selected not only for their experience but also for their specific roles, which deeply involve working with requirements.

6.2. Construct validity

Construct validity is concerned with the degree to which our measurements indeed reflect what we claim to measure [56]. From this perspective, construct validity threats may arise during both interview and survey. Regarding the interview, a potential threat lies in the risk of participants misinterpreting the interview questions and us misinterpreting their responses. To address this, we used a semi-structured interview protocol, which allowed clarification of misunderstandings on proposed questions and facilitated the investigation of third factors that may affect given answers [57]. As a first step, we allowed the interviewees to discuss issues they perceived as important or frequent in their day-to-day requirements analysis activities. This approach mitigated the researchers' bias, i.e., that we did not impose or bias their perspectives. Thus, interviewees were encouraged to express their experiences and thoughts using their own terminology, even if not entirely precise or aligned with established requirements engineering frameworks. As a second step, after this open-ended discussion, we introduced the concept of requirement smells as defined in the literature. While some interviewees were already familiar with this concept, others were not. The practical experience of our interviewees in requirements analysis enabled them to refine their initial inputs and map their concepts onto the proposed smells. Furthermore, differences in perception emerged based on the interviewees' organizational roles, which sometimes influenced their prioritization or interpretation of specific smells. For instance, certain interviewees challenged aspects of the proposed categorization, such as the separation of "Non-Understandable" and "Not-Traceable", arguing that these were secondary effects of other smells like Ambiguity or Non-verifiability. Such feedback underscored the need for a more stringent classification system, leading to valuable insights and the consideration of established standards like INCOSE [17] (which has been used later for the survey). By minimizing interviewer bias and allowing practitioners to reflect on their own experiences freely, we ensured that the findings authentically represent the diversity of perspectives across roles and expertise, enhancing the credibility of our study.

Concerning the survey, we provided a "Comments" section for each question to let the participants refine their answers or better argue their choices.

As remarked by [37], other possible threats might be represented by bias induced by poor-quality questions or by specific type of scale adoption, i.e. double-barrelled questions, pattern answering, anchoring or intrinsic distractions. To minimize this type of threat, we designed each question to focus on a single, specific smell at a time. Specifically, we adopted the following pattern: Smell Definition - Introduction - Question. The Smell Definition and Introduction were added to re-focus the participant's attention on the specific smell. Afterwards, each question was phrased with the formula 'Regardless of, and not limited to, the above introduction...", reported in bold and greater font size to limit possible anchoring-to-introduction effect. Moreover, to minimize the possibility of "pattern answering", we randomly shuffled the order of metrics.

Concerning those questions where the Likert scale technique has been applied, i.e. F2, F4, F5, F6, we believe that "pattern answering" is unlikely due to the small number of questions (i.e., only 4 questions).

Concerning the questions where the 100-point distribution technique has been applied, we acknowledge that tracking the partial sum of allocated points might potentially distract participants from correct points allocation [37]. Therefore, we implemented a running total display of assigned points, aiding participants in focusing on the question itself.

6.3. External validity

External validity is concerned with the extent to which the research elements (subjects, artefacts, etc.) are representative of actual elements [56]. The external validity of our findings is limited by the context of a single company. However, we expect our results to apply to safety–critical contexts, i.e., where requirements are numerous, frequently updated, and systematically tracked over the entire product lifecycle.

While we aimed for a representative sample, results should be generalized with caution. To address this limitation, participants were drawn from diverse roles and departments, as reflected in the demographic data, strengthening the generalizability of the survey results.

Finally, we emphasize that the six Findings presented are not tied to specific smells but instead reflect broader trends. Both interviews and surveys used distinct categorizations but converged on consistent findings. For instance, the takeaway from F3 is that the frequency changes across smells, rather than the relative frequency of any particular smell.

6.4. Reliability validity

Reliability validity concerns to what extent the data and the analysis are dependent on the specific researchers [57]. Concerning the reliability of data coming from the Interview, data has already been partially discussed in the Construct validity section. Possible reliability threats may arise from the analysis of interview transcription. To mitigate any possible dependency on researchers, we ensured that all interviews were audio-recorded [57], allowing for detailed and accurate transcription and analysis. Each author independently reviewed the recorded interviews and participants' responses to capture initial interpretations without bias. Subsequently, these independent analyses were verified collaboratively to ensure consistency in four review sessions to resolve discrepancies, ensure consistency, and verify that no key insights were overlooked or misinterpreted [58].

6.5. Conclusion validity

Conclusion validity concerns issues that affect the ability to draw accurate conclusions regarding the observed relationships between the independent and dependent variables [56]. A potential threat to conclusion validity lies in the use of statistical methods that may be sensitive to violations of underlying assumptions, such as normality or independence of observations. To mitigate this threat, we employed a rank-based linear mixed model (RBLMM) and Wilcoxon Signed-Rank (WSR) tests, both of which are nonparametric methods robust to deviations from normality. Additionally, the inclusion of survey respondents' IDs as a random effect in the RBLMM accounted for within-subject variability, ensuring that the repeated-measures nature of the data was properly handled. By setting a conservative alpha value of 0.01, we further reduced the risk of Type I errors due to the multiple comparisons conducted in the analysis.

7. Conclusion

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. To the best of our knowledge, no previous study analysed how frequency, severity, or effects vary across types of smells.

We interview ten experienced practitioners from different divisions of a large international company in the safety–critical domain called MBDA Italy Spa. Then we survey 58 people from the same company to support our findings and extend the analysis to metrics for measuring specific types of requirements smells effect.

Our results show that the smell types perceived as most severe are Ambiguity and Unverifiability, while the most frequent are Ambiguity and Incompleteness. We also provide a set of six findings about requirements for smells, such as that the effects of smells are expected to differ across smell types and stages of the project. As expected, the survey supported our interview results largely but not completely. Possible reasons for a large but incomplete agreement include the diverse contexts and roles represented in the survey compared to the smaller, more focused group of interviewees. Thus, the small disagreement highlights the value of combining qualitative and quantitative approaches, as they reveal how smells are perceived across different roles and experiences.

Our results show that focusing on the quality of early requirements is crucial. Ensuring that requirements documentation is smell-free can minimize rework in later development stages. In other words, by investing in the early detection and removal of requirements smells, organizations can reduce the costs associated with later-stage rework and bug fixing. Since our results show the effects of smells vary across smells, the targeted monitoring of specific metrics is essential.

The development teams can quickly identify and correct the issues by focusing on the metrics most relevant to the interesting smells, such as the number of queries for ambiguous requirements. This customized approach allows for effective problem-solving and the ability to keep projects moving forward.

In addition, the communication and coordination of stakeholders are crucial to eliminate confusion and ensure that all stakeholders understand the requirements. Improved communication can help avoid confusion and guarantee that all participants are on the same page; hence, the chances of defects arising from misinterpreted requirements will be minimized. These high-risk areas should be the first to be addressed by mitigation strategies, as this will ensure that potential problems are identified and resolved before they become serious enough to affect project stability and quality.

Our results also support classifying code smells according to their potential impact. Researchers can also create taxonomies that organize smells based on their consequences, such as rework, bugs, or usability problems. This classification makes it possible to develop specific mitigation or validation strategies to deal with the issues created by each type of smell. By understanding the specific impacts of different smells, teams can implement more effective and focused interventions, ultimately improving software projects' overall quality and success.

Our paper paves the way to future works in many dimensions:

1. *Prove 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 [59] 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. Conversely, 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 such as controlled experiments measuring the size of smell effects.
2. *Extend over different contexts*. Performing longitudinal studies across multiple organizations or industries would allow for the generalization of findings. Such studies could observe how teams address specific requirement smells over time and how their mitigation strategies adapt and evolve with accumulated experience. Moreover, our industrial context features requirements written in natural language; it would be interesting to replicate this study over other types of requirements.

3. *Tool development*. We envision future types of static analysers that in addition to the current provided feature of identifying smells, could also automatically measure the effects of smells by integrating it with control versions systems like Git and issue tracking systems like Jira.
4. *Use of Large Language Model (LLM)*. Future research could explore how LLMs enhance the requirements engineering process. For instance, we could investigate how LLMs can assist in writing clear and unambiguous requirements, identifying inconsistencies, and ensuring alignment with industry standards. The investigation of the use of LLM is interesting since they are sensible to the training data, which needs to be related to the specific industry context, and their use can provide inconsistent responses, generating information that might not be accurate and using language that could be unclear or ambiguous.

CRediT authorship contribution statement

Emanuele Gentili: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Davide Falessi:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Interview questions

The list of questions is hereafter reported:

Interviewee characterization

1. What is your current role? (Role)
    (a) SW Engineer (SE): designs, develops, and maintains software systems for various applications.
    (b) Junior SW Engineer (JSE): same as SW Engineer, but with limited domain and processes knowledge.
    (c) SW Team Leader (STL): leads a small-size software team inside a SW project, and ensures successful delivery of high-quality software solutions.
    (d) SW Technical Expert (STX): provides specialized knowledge and expertise in specific software domains.
    (e) SW Architect (SA): provides specialized knowledge and expertise in designing and overseeing the high-level structure and behaviour of software systems, ensuring they meet functional and performance requirements.
    (f) SW Project Leader (SPL): leads a software project, coordinates teams, manages resources, and ensures successful delivery of high-quality software solutions.
    (g) 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.
    (h) 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)

Project characterization

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)
    (a) 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.
    (b) 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 Lines Of Code does a software component consist of on average? (#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?
5. What are the metrics that could be used to measure the effects that a certain smell could induce on project development?

Appendix B. Survey questions

The list of questions is hereafter reported:

Introduction

Welcome to our research survey on requirement smells, conducted by SW Eng. Emanuele Gentili, MBDA, and Prof. Davide Falessi, University of Rome Tor Vergata. In software development, a "software requirement smell" refers to a set of characteristics or indicators of a requirement specification that suggest potential issues or weaknesses in the requirement itself. The concept of "requirement smell" is analogous to "code smell", which refers to specific patterns or structures in source code that might indicate underlying problems or the need for refactoring. Software requirement smells are not errors or bugs by themselves, but they warn that the requirement may lack clarity, be poorly written, or suffer from other problems that could impact the software project's success. Identifying and addressing requirements early in the development process is essential for improving the quality of the requirement documentation and reducing the risk of misinterpretation or miscommunication among stakeholders and development teams. From an industrial perspective, we want to focus our time and effort on preventing the requirement smells of high interest. The study aims to identify the requirement smells perceived as insidious and dangerous for project development and their connections with specific aspects of the software development lifecycle, like Design, Implementation, Testing and Maintenance. Therefore, **there are no correct or incorrect answers**; thus, you are invited to provide your opinion without any kind of repercussion. There are no risks associated with participation in this study. Your participation will take approximately 30 min. Your answers will be kept confidential and only reported in an aggregated form. The survey participation deadline is YYYY-MM-DD If you have questions regarding this study or would like to be informed of the results when the study is completed, please feel free to contact Eng. Emanuele Gentili at emanuele.gentili@mbda.it. If you agree to participate in this research project voluntarily, please indicate your agreement by completing and submitting the following questionnaire. Feel free to print a copy of this consent form for your records, and thank you for participating in this research!

1. I consent to allow my answers to be used anonymously in this study for research purposes.

    (a) Yes
    (b) No

Demographics

1. Which roles have you played (not limited to MBDA experience)? (MC)

    (a) SW Engineer (SE)
    (b) Junior SW Engineer (JSE)
    (c) SW Team Leader (STL)
    (d) SW Technical Expert (STX)
    (e) SW Architect (SA)
    (f) SW Project Leader (SPL)
    (g) SW Group Leader (SGL)
    (h) Head of Department (HOD)
    (i) Other (please specify)

2. How many years of experience do you have as a professional?
3. Which departments have you been involved with? (MC)

    (a) Ground or Naval
    (b) Embedded
    (c) Missile
    (d) Equipment

4. Have you ever worked in Hard Real-Time context? (SC)

    (a) Yes
    (b) No

5. Which types of requirements have you ever been involved with? [15] (MC)

    (a) Functional/Performance requirements
    (b) Interface requirements
    (c) Process requirements
    (d) Quality (Non-functional) requirements
    (e) Usability/Quality-in-use requirements
    (f) Human Factors requirements

Agreement with findings

- **Glossary**
  The characteristics that a requirement (or a set of requirements) shall possess according to the standard *ISO/IEC/IEEE 29148* are:

    1. **Necessary**. The requirement defines an essential capability, characteristic, constraint and/or quality factor. If it is not included in the set of requirements, a deficiency in capability or characteristic will exist, which cannot be fulfilled by implementing other requirements. The requirement is currently applicable and has not been made obsolete by the passage of time. Requirements with planned expiration dates or applicability dates are clearly identified. Its absence causes the smell called *Unnecessary*.
    2. **Appropriate**. The specific intent and amount of detail of the requirement is appropriate to the level of the entity to which it refers (level of abstraction appropriate to the level of entity). This includes avoiding unnecessary constraints on the architecture or design while allowing implementation independence to the extent possible. Its absence causes the smell called *Inappropriate*.
    3. **Unambiguous**. The requirement is stated in such a way so that it can be interpreted in only one way. The requirement is stated simply and is easy to understand. Its absence causes the smell called *Ambiguous*.
    4. **Complete**. The requirement sufficiently describes the necessary capability, characteristic, constraint or quality factor to meet the entity need without needing other information to understand the requirement. Its absence causes the smell called *Incomplete*.
    5. **Singular**. The requirement states a single capability, characteristic, constraint or quality factor. Although a single requirement consists of a single function, quality or constraint, it can have multiple conditions under which the requirement is to be met. Its absence causes the smell called *Plural* (Non-singular).
    6. **Feasible**. The requirement can be realized within system constraints (e.g., cost, schedule, technical) with acceptable risk. Its absence causes the smell called *Infeasible*.
    7. **Verifiable**. The requirement is structured and worded such that its realization can be proven (verified) to the customer's satisfaction at the level the requirements exist. Verifiability is enhanced when the requirement is measurable. Its absence causes the smell called *Unverifiable*.
    8. **Correct**. The requirement is an accurate representation of the entity need from which it was transformed. Its absence causes the smell called *Incorrect*.
    9. **Consistent**. The set of requirements contains individual requirements that are unique, do not conflict with or overlap with other requirements in the set, and the units and measurement systems are homogeneous. The terminology used within the set of requirements is consistent, i.e. the same term is used throughout the set to mean the same thing. Its absence causes the smell called *Inconsistent*.
    10. **Conforming**. The individual items conform to an approved standard template and style for writing requirements, when applicable. Its absence causes the smell called *Non-Conforming*.

- **Finding 1**: *The perceived severity varies across types of smells*.

    – Please distribute 100 points across the 10 smells below according to their **severity**, i.e., how much you think it is important to avoid them in requirements. A higher score indicates a higher severity.

        1. Unnecessary
        2. Inappropriate
        3. Ambiguous
        4. Incomplete
        5. Plural (Non-singular)
        6. Infeasible
        7. Unverifiable
        8. Incorrect
        9. Inconsistent
        10. Non-conforming

    – Comments

- **Finding 3**: *The perceived frequency varies across types of smells*.

    – Please distribute 100 points across the 10 smells below according to their **frequency**, i.e., how much you usually find them in requirements. A higher score indicates a higher frequency.

        1. Unnecessary
        2. Inappropriate

3. Ambiguous
4. Incomplete
5. Plural (Non-singular)
6. Infeasible
7. Unverifiable
8. Incorrect
9. Inconsistent
10. Non-conforming

– Comments

- **Finding 2**: *The perceived severity of the same smell varies across project domains*.

  – We know that requirement smells can cause rework and time and cost overruns; in some cases, a smell might even be catastrophic. For instance, requirements concerning the system's performance are critical for Hard Real Time (HRT) systems. A single smell in the performance requirement of an HRT system might significantly impact the overall project or even people's lives. For instance, regarding the *Unverifiable* smell, if a performance requirement does not come with explicit time constraints, we have a significant 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.
  Please provide the level of agreement with it.

    1. Strongly disagree
    2. Disagree
    3. Neutral or I do not know
    4. Agree
    5. Strongly agree

  – Comments

- **Finding 4**: *The frequency of a smell is perceived differently across roles and 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".
  Please provide the level of agreement with it.

    1. Strongly disagree
    2. Disagree
    3. Neutral or I do not know
    4. Agree
    5. Strongly agree

  – Comments

- **Finding 5**: *The severity of a smell might change across roles or phases.*.

  – We know that requirement smells can have negative effects on software development; however, we realize that the effects of smells might be null or even positive in some circumstances. Let us have an example of how the effect of the "underspecification", a sub-smell of the *Incompleteness* 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 better understand what they want. 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 better specify their needs. Conversely, 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 requirement at an early stage can cause the SW architect to design an incorrect architecture with small flexibility and an inability to satisfy constraints that will come up later during the development stage. So, it can potentially bring the redesign of the whole architecture at the price of losing time and money and increasing the frustration of the whole development team. On the other side, a too-abstract requirement is not necessarily bad news: from Project Leader and Technical Expert's 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. We do not know if this reasoning applies to requirements sub-smells other than "underspecification".
  Please provide the level of agreement with it.

    1. Strongly disagree
    2. Disagree
    3. Neutral or I do not know
    4. Agree
    5. Strongly agree

  – Comments

- **Finding 6**: *The perceived effects may vary across types of smells*.

  – We know that requirements smells can cause rework and time and cost overruns. Moreover, reasonably, specific smells cause specific problems. However, to the best of our knowledge, the specific effects of specific requirement smells are unknown. Let us discuss examples of the *Verifiability*, *Ambiguity*, and a combination of *Inappropriateness* and *Non-singularity* 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 to provide evidence of the desired behaviour) and during the Testing phase (in fact the number and complexity of Test Cases can grow significantly)... and a poorly tested SW is likely to exhibit bugs during the Maintenance phase, leading to a high impact in term of rework, time, extra costs and customer

satisfaction, with a general loss of credibility of the Company. Regarding *Ambiguity*, if a requirement is unclear, this will likely need to go back and forth between requirements engineers and developers to identify and formalize 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. Finally, a combination of *Inappropriate* and *Plural* smells might lead to a complex requirement: it could be overspecified or underspecified (*Inappropriate*), and it could describe several capabilities with multiple conditions to be met (*Non-singularity*). And, 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. 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.
Please provide the level of agreement with it.

1. Strongly disagree
2. Disagree
3. Neutral or I do not know
4. Agree
5. Strongly agree

– Comments

Smell-to-effect linking

In this section, we want to explore the connection between a specific requirement smells and its measurable effects on a project. A list of possible metrics is reported hereafter, but please feel free to add others at your convenience.

List of Metrics

- **M1**: High number of Defects - i.e. bugs during the software development phase
- **M2**: High number of Queries - i.e. requests of clarification about requirements content
- **M3**: High number of Change Requests of the TRS (Technical Requirement Specification) document
- **M4**: High number of code Commits
- **M5**: High number of LOC (lines of code) tested by inspection
- **M6**: High number of Line of Code impacted by Safety Critical standards
- **M7**: Low score for code Quality Assurance - e.g. the presence of duplicate code, high cyclomatic complexity, deep nesting, long methods, inappropriate comments, etc...
- **M8**: High number of Technical Facts - i.e software bugs in production, coming after a real deployment of a system on the field
- **M9**: High number of document updates - i.e. any documents produced during the SW development lifecycle
- **M10**: High number of test updates/reworks
- **M11**: Delay in SRS (Software Requirement Specification) document definition
- **M12**: Delay in starting coding activities
- **M13**: Others — Please specify in Comments

Unnecessary

- Definition: the requirement pertains to a capability, characteristic, constraint, or quality factor that is not essential, and its absence does not result in a deficiency in capability or characteristic.
- Introduction: excluding dispensable requirements might be seen as beneficial, but it can be problematic as some seemingly non-essential elements play a crucial role in overall system stability and functionality. Omitting these may lead to unforeseen deficiencies, hindering the system's ability to adapt, perform optimally, or meet user expectations. In essence, what appears dispensable on the surface might hold significance for the system's robustness and user satisfaction.
- Question: **regardless of, and not limited to**, the above introduction, what metrics could measure the negative impact of the "*Unnecessary*" requirement smell?
  Please choose from **List of Metrics** (B)
- Comments

Inappropriate

- Definition: the specific intent and level of detail of the requirement are unsuitable for the level of the entity to which it refers, introducing unnecessary constraints on the architecture or design.
- Introduction: inappropriate requirements pose challenges as insufficient detailing may result in misunderstandings among stakeholders, leading to a misalignment of expectations. This lack of precision can hinder effective communication, potentially causing errors in system design or implementation. Inappropriately specified requirements may impede the development process, impacting the overall quality of the final product.
- Question: **regardless of, and not limited to**, the above introduction, what metrics could measure the negative impact of the *Inappropriate* requirement smell?
  Please choose from **List of Metrics** (B)
- Comments

Ambiguous

- Definition: the requirement is stated in a way that allows for multiple interpretations, and it may be unclear or open to different meanings.
- Introduction: ambiguous requirements are problematic as they introduce uncertainty, leading to varied interpretations among developers and stakeholders. This ambiguity can result in misunderstandings, potentially causing divergent implementations and compromising the system's intended functionality.
- Question: **regardless of, and not limited to**, the above introduction, what metrics could measure the negative impact of the *Ambiguous* requirement smell?
  Please choose from **List of Metrics** (B)
- Comments

Incomplete

- Definition: the requirement lacks sufficient description to meet the entity's need, requiring additional information to understand it adequately.
- Introduction: incomplete requirements are problematic as they leave critical aspects undefined, leading to gaps in system functionality. This can hinder developers' understanding, potentially resulting in inadequate system design or implementation. Incompleteness may lead to a final product that fails to meet user expectations and operational needs.

- Question: **regardless of, and not limited to**, the above introduction, what metrics could measure the negative impact of the *Incomplete* requirement smell?
  Please choose from **List of Metrics** (B)
- Comments

Plural

- Definition: the requirement states multiple capabilities, characteristics, constraints, or quality factors instead of a single one.
- Introduction: a plural requirement can be problematic as it introduces complexity by bundling multiple aspects in a single statement, making it challenging to prioritize and implement effectively. This complexity may lead to conflicting priorities, confusion among development teams, and potential errors in the system design or implementation process.
- Question: **regardless of, and not limited to**, the above introduction, what metrics could measure the negative impact of the *Plural* requirement smell?
  Please choose from **List of Metrics** (B)
- Comments

Infeasible

- Definition: the requirement cannot be realized within system constraints, presenting unacceptable risk, or exceeding limits in areas such as cost, schedule, or technical aspects.
- Introduction: an infeasible requirement is problematic as it sets unrealistic expectations, potentially straining resources and leading to project delays or failure. This may result in increased costs, technical challenges, and dissatisfaction among stakeholders. Ensuring feasibility in requirements is essential for realistic project planning and successful system development.
- Question: **regardless of, and not limited to**, the above introduction, what metrics could measure the negative impact of the *Infeasible* requirement smell?
  Please choose from **List of Metrics** (B)
- Comments

Unverifiable

- Definition: the requirement is not structured or worded in a way that allows for its realization to be proven or verified to the customer's satisfaction.
- Introduction: an unverifiable requirement is problematic as it lacks a clear verification path, hindering the ability to prove its realization to the customer's satisfaction. This uncertainty can lead to challenges in quality assurance, potentially resulting in unvalidated system functionalities and eroding stakeholder trust. Verifiability is crucial for ensuring a robust and reliable system development process.
- Question: **regardless of, and not limited to**, the above introduction, what metrics could measure the negative impact of the *Unverifiable* requirement smell?
  Please choose from **List of Metrics** (B)
- Comments

Incorrect

- Definition: the requirement does not accurately represent the entity need from which it was transformed, introducing inaccuracies or errors.
- Introduction: an incorrect requirement is problematic as it misrepresents the user's actual needs, leading to misguided development efforts and potential dissatisfaction. Inaccuracy in requirements can result in a system that fails to align with user expectations, impacting usability, functionality, and overall user satisfaction. Ensuring correctness in requirements is essential for building a system that effectively meets user needs.
- Question: **regardless of, and not limited to**, the above introduction, what metrics could measure the negative impact of the *Incorrect* requirement smell?
  Please choose from **List of Metrics** (B)
- Comments

Inconsistent

- Definition: the set of requirements contains conflicting or overlapping individual requirements, and the units and measurement systems are not homogeneous. The terminology used within the set is inconsistent.
- Introduction: inconsistent requirements pose problems as they introduce confusion and potential conflicts in system development. Lack of coherence may lead to miscommunication among development teams, hindering collaboration and causing integration issues. Ensuring consistency in requirements is vital for a smooth development process and the creation of a reliable and cohesive system.
- Question: **regardless of, and not limited to**, the above introduction, what metrics could measure the negative impact of the *Inconsistent* requirement smell?
  Please choose from **List of Metrics** (B)
- Comments

Non-Conforming

- Definition: the individual items do not adhere to an approved standard template and style for writing requirements, when applicable.
- Introduction: nonconforming requirements are problematic as they deviate from established standards, leading to inconsistencies and potential interoperability issues. Non-compliance may result in a product that fails to meet industry norms, compromising quality, reliability, and overall project success. Adhering to approved standards is crucial for ensuring the integrity and compatibility of the final system.
- Question: **regardless of, and not limited to**, the above introduction, what metrics could measure the negative impact of the *Non-conforming* requirement smell?
  Please choose from **List of Metrics** (B)
- Comments

Data availability

No data was used for the research described in the article.

References

[1] K. Pohl, Requirements Engineering - Fundamentals, Principles, and Techniques, Springer, 2010, URL: http://www.springer.com/computer/swe/book/978-3-642-12577-5?changeHeader.
[2] M. Kassab, C. Neill, P. Laplante, State of practice in requirements engineering: contemporary data, Innov. Syst. Softw. Eng. 10 (2014) 235–241.
[3] A. Mavin, P. Wilkinson, Big ears (the return of "easy approach to requirements engineering"), in: 2010 18th IEEE International Requirements Engineering Conference, IEEE, 2010, pp. 277–282.
[4] J.J. Ahonen, P. Savolainen, Software engineering projects may fail before they are started: Post-mortem analysis of five cancelled projects, J. Syst. Softw. 83 (11) (2010) 2175–2187.
[5] M.E.C. Hull, K. Jackson, J. Dick (Eds.), Requirements Engineering, third ed., Springer, 2011, http://dx.doi.org/10.1007/978-1-84996-405-0.
[6] A. Veizaga, S.Y. Shin, L.C. Briand, Automated smell detection and recommendation in natural language requirements, 2023, http://dx.doi.org/10.48550/arXiv.2305.07097, CoRR abs/2305.07097.
[7] L. Montgomery, D. Fucci, A. Bouraffa, L. Scholz, W. Maalej, Empirical research on requirements quality: a systematic mapping study, in: G. Engels, R. Hebig, M. Tichy (Eds.), Software Engineering 2023, Fachtagung des GI-Fachbereichs Softwaretechnik, 20.-24. Februar 2023, Paderborn, in: LNI, vol. P-332, Gesellschaft für Informatik e.V., 2023, pp. 91–92, URL: https://dl.gi.de/20.500.12116/40098.

[8] M. Kretsou, E. Arvanitou, A. Ampatzoglou, I.S. Deligiannis, V.C. Gerogiannis, Change impact analysis: A systematic mapping study, J. Syst. Softw. 174 (2021) 110892, http://dx.doi.org/10.1016/j.jss.2020.110892.

[9] D.M. Fernández, S. Wagner, M. Kalinowski, M. Felderer, P. Mafra, A. Vetrò, T. Conte, M. Christiansson, D. Greer, C. Lassenius, T. Männistö, M. Nayabi, M. Oivo, B. Penzenstadler, D. Pfahl, R. Prikladnicki, G. Ruhe, A. Schekelmann, S. Sen, R.O. Spínola, A. Tuzcu, J.L. de la Vara, R.J. Wieringa, Naming the pain in requirements engineering - Contemporary problems, causes, and effects in practice, Empir. Softw. Eng. 22 (5) (2017) 2298–2338, http://dx.doi.org/10.1007/s10664-016-9451-7.

[10] H. Femmer, D.M. Fernández, S. Wagner, S. Eder, Rapid quality assurance with requirements smells, 2016, CoRR abs/1611.08847, URL: http://arxiv.org/abs/1611.08847.

[11] E. Gentili, D. Falessi, Characterizing requirements smells, in: International Conference on Product-Focused Software Process Improvement, Springer, 2023, pp. 387–398.

[12] I. Sommerville, P. Sawyer, Requirements Engineering: A Good Practice Guide, John Wiley & Sons, Inc., 1997.

[13] S. Robertson, J. Robertson, Mastering the Requirements Process: Getting Requirements Right, Addison-wesley, 2012.

[14] J. Dick, M.E.C. Hull, K. Jackson, Requirements Engineering, fourth ed., Springer, 2017, http://dx.doi.org/10.1007/978-3-319-61073-3.

[15] ISO/IEC/IEEE international standard - Systems and software engineering – Life cycle processes – Requirements engineering, 2018, pp. 1–104, http://dx.doi.org/10.1109/IEEESTD.2018.8559686, ISO/IEC/IEEE 29148: 2018(E).

[16] INCOSE, INCOSE Systems Engineering Handbook, John Wiley & Sons, 2023.

[17] INCOSE, Guide to writing requirements, 2023, https://www.incose.org/docs/default-source/working-groups/requirements-wg/gtwr/incose_rwg_gtwr_v4_040423_final_drafts.pdf.

[18] K. Subramaniam, D. Liu, B.H. Far, A. Eberlein, UCDA: Use case driven development assistant tool for class model generation, in: F. Maurer, G. Ruhe (Eds.), Proceedings of the Sixteenth International Conference on Software Engineering & Knowledge Engineering, SEKE'2004, Banff, Alberta, Canada, June 20-24, 2004, 2004, pp. 324–329.

[19] V. Mencl, Deriving Behavior Specifications from Textual Use Cases, Citeseer, 2004.

[20] A. Ferrari, G. Gori, B. Rosadini, I. Trotta, S. Bacherini, A. Fantechi, S. Gnesi, Detecting requirements defects with NLP patterns: an industrial experience in the railway domain, Empir. Softw. Eng. 23 (6) (2018) 3684–3733, http://dx.doi.org/10.1007/s10664-018-9596-7.

[21] J. Frattini, Identifying relevant factors of requirements quality: An industrial case study, in: International Working Conference on Requirements Engineering: Foundation for Software Quality, Springer, 2024, pp. 20–36.

[22] J. Frattini, L. Montgomery, J. Fischbach, D. Mendez, D. Fucci, M. Unterkalmsteiner, Requirements quality research: a harmonized theory, evaluation, and roadmap, Requir. Eng. 28 (4) (2023) 507–520.

[23] S. Wagner, K. Lochmann, S. Winter, F. Deissenboeck, E. Juergens, M. Herrmannsdoerfer, L. Heinemann, M. Kläs, A. Trendowicz, J. Heidrich, et al., The quamoco quality meta-model, 2012.

[24] P. Kruchten, What do software architects really do? J. Syst. Softw. 81 (12) (2008) 2413–2416, http://dx.doi.org/10.1016/j.jss.2008.08.025.

[25] D. Falessi, G. Cantone, R. Kazman, P. Kruchten, Decision-making techniques for software architecture design: A comparative survey, ACM Comput. Surv. 43 (4) (2011) 33:1–33:28, http://dx.doi.org/10.1145/1978802.1978812.

[26] P.A. Laplante, et al., Real-Time Systems Design and Analysis, Wiley New York, 2004.

[27] B.W. Boehm, Software engineering economics, IEEE Trans. Softw. Eng. 10 (1) (1984) 4–21, http://dx.doi.org/10.1109/TSE.1984.5010193.

[28] A.M. Davis, Software Requirements - Analysis and Specification, Prentice Hall, 1990.

[29] Y. Seki, S. Hayashi, M. Saeki, Detecting bad smells in use case descriptions, in: D.E. Damian, A. Perini, S. Lee (Eds.), 27th IEEE International Requirements Engineering Conference, RE 2019, Jeju Island, Korea (South), September 23-27, 2019, IEEE, 2019, pp. 98–108, http://dx.doi.org/10.1109/RE.2019.00021.

[30] A. Veizaga, M. Alférez, D. Torre, M. Sabetzadeh, L.C. Briand, On systematically building a controlled natural language for functional requirements, Empir. Softw. Eng. 26 (4) (2021) 79, http://dx.doi.org/10.1007/s10664-021-09956-6.

[31] A.M. Davis, Ó.D. Tubío, A.M. Hickey, N.J. Juzgado, A.M. 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, IEEE Computer Society, 2006, pp. 176–185, http://dx.doi.org/10.1109/RE.2006.17.

[32] B. Solemon, S. Sahibuddin, A.A.A. Ghani, Requirements engineering problems and practices in software companies: An industrial survey, in: D. Slezak, T. Kim, K. Akingbehin, T. Jiang, J.M. Verner, S.A. ao (Eds.), 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, in: Communications in Computer and Information Science, vol. 59, Springer, 2009, pp. 70–77, http://dx.doi.org/10.1007/978-3-642-10619-4_9.

[33] W. Schofield, Survey sampling, Data Collect. Anal. 2 (2006) 332.

[34] C. Robson, Real world research blackwell, 2002, 2º Edişão.

[35] E. Kang, H.-J. Hwang, The Importance of Anonymity and Confidentiality for Conducting Survey Research.

[36] T. Nemoto, D. Beglar, Likert-scale questionnaires, in: JALT 2013 Conference Proceedings, 2014, pp. 1–8.

[37] I. Brace, Questionnaire Design: How to Plan, Structure and Write Survey Material for Effective Market Research, Kogan Page Publishers, 2018.

[38] J.A. Khan, I.U. Rehman, Y.H. Khan, I.J. Khan, S. Rashid, Comparison of requirement prioritization techniques to find best prioritization technique, Int. J. Mod. Educ. Comput. Sci. 7 (11) (2015) 53.

[39] T. Baguley, Serious Stat: A Guide to Advanced Statistics for the Behavioral Sciences, Bloomsbury publishing, 2018.

[40] F. Wilcoxon, Individual comparisons by ranking methods, in: Breakthroughs in Statistics: Methodology and Distribution, Springer, 1992, pp. 196–202.

[41] D.C. Montgomery, G.C. Runger, Applied Statistics and Probability for Engineers, John wiley & sons, 2010.

[42] M. Greenacre, Correspondence Analysis in Practice, chapman and hall/crc, 2017.

[43] E.M. Redmiles, Y. Acar, S. Fahl, M.L. Mazurek, A summary of survey methodology best practices for security and privacy researchers, 2017.

[44] D.G. Altman, J.M. Bland, Statistics notes: Absence of evidence is not evidence of absence, Bmj 311 (7003) (1995) 485.

[45] B. Intrigila, G. Della Penna, A. D'Ambrogio, D. Campagna, M. Grigore, Process-oriented requirements definition and analysis of software components in critical systems, Computers 12 (9) (2023) 184.

[46] D.G. Firesmith, A taxonomy of safety-related requirements, in: International Workshop on High Assurance Systems, RHAS'05, Citeseer, 2005.

[47] M.A. Rauf, S. Bibi, S. Ali, T. AlSaedi, S. Ur Rehman, K. Mahmood, M. Kundi, A cost effective communication model for requirements elicitation in global software development, Sci. Rep. 13 (1) (2023) 18730.

[48] F.N.J. Muhamad, S.H. Ab Hamid, H. Subramaniam, R. Abdul Rashid, F. Fahmi, Fault-prone software requirements specification detection using ensemble learning for edge/cloud applications, Appl. Sci. 13 (14) (2023) 8368.

[49] D.M. Berry, E. Kamsties, Ambiguity in requirements specification, in: Perspectives on Software Requirements, Springer, 2004, pp. 7–44.

[50] P. Achimugu, A. Selamat, R. Ibrahim, M.N. Mahrin, A systematic literature review of software requirements prioritization research, Inf. Softw. Technol. 56 (6) (2014) 568–585.

[51] J. Biolchini, P.G. Mian, A.C.C. Natali, G.H. Travassos, Systematic review in software engineering, Syst. Eng. Comput. Sci. Dep. COPPE/UFRJ, Tech. Rep. ES 679 (05) (2005) 45.

[52] M.V. Mäntylä, C. Lassenius, Subjective evaluation of software evolvability using code smells: An empirical study, Empir. Softw. Eng. 11 (2006) 395–431.

[53] G. Bavota, A. Qusef, R. Oliveto, A. De Lucia, D. Binkley, Are test smells really harmful? an empirical study, Empir. Softw. Eng. 20 (2015) 1052–1094.

[54] A. Yamashita, L. Moonen, To what extent can maintenance problems be predicted by code smell detection?–An empirical study, Inf. Softw. Technol. 55 (12) (2013) 2223–2242.

[55] A. Davis, S. Overmyer, K. Jordan, J. Caruso, F. Dandashi, A. Dinh, G. Kincaid, G. Ledeboer, P. Reynolds, P. Sitaram, et al., Identifying and measuring quality in a software requirements specification, in: [1993] Proceedings First International Software Metrics Symposium, Ieee, 1993, pp. 141–152.

[56] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, A. Wesslén, et al., Experimentation in Software Engineering, vol. 236, Springer, 2012.

[57] P. Runeson, M. Höst, Guidelines for conducting and reporting case study research in software engineering, Empir. Softw. Eng. 14 (2009) 131–164.

[58] M.Q. Patton, Qualitative Research & Evaluation Methods: Integrating Theory and Practice, Sage publications, 2014.

[59] D.I.K. Sjøberg, A.F. Yamashita, B.C.D. Anda, A. Mockus, T. Dybå, Quantifying the effect of code smells on maintenance effort, IEEE Trans. Softw. Eng. 39 (8) (2013) 1144–1156, http://dx.doi.org/10.1109/TSE.2012.89.

Emanuele Gentili is a Ph.D. student in Computer Science at the University of Rome Tor Vergata, Italy, and currently V & V Team Leader at MBDA Italy S.p.a. His main research interest is artificial intelligence applied to requirements engineering for improving the design and verification of complex systems. He received his MSc and BSc degrees in Automation and Robotics Engineering from the University of Rome Tor Vergata, Italy.

Davide Falessi is an Associate Professor of Software Engineering at the University of Rome Tor Vergata, Italy. He is the Associate Editor in Software Economics of IEEE Software and an Editorial Board member of the Empirical Software Engineering Journal. His main research interest is in devising and empirically assessing scalable solutions for developing software-intensive systems. He received his Ph.D., MSc, and BSc degrees in Computer Engineering from the University of Rome Tor Vergata, Italy.