Revista Facultad de Ingeniería
Journal Homepage: https://revistas.uptc.edu.co/index.php/ingenieria

Systematic Mapping of the Literature on Smells in Software Development Requirements

Mayra-Alejandra Castillo-Motta[1]
Rubén-Darío Dorado-Córdoba[2]
César-Jesús Pardo-Calvache[3]
Carlos-Eduardo Orozco-Garcés[4]

Received: August 19, 2022 **Accepted:** January 26, 2023 **Published:** February 09, 2023

Citation: M.-A. Castillo-Motta, R.-D. Dorado-Córdoba, C.-J. Pardo-Calvache, C.-E. Orozco-Garcés, "Systematic Mapping of the Literature on Smells in Software Development Requirements," *Revista Facultad de Ingeniería*, vol. 32 (63), e15233, 2023. https://doi.org/10.19053/01211129.v32.n63.2023.15233

Abstract

One of the activities responsible for the success of a software development project is the specification of requirements, whose purpose is to ensure that the wishes or

[1] Universidad del Cauca (Pitalito-Huila, Colombia). mayracastillo@unicauca.edu.co. ORCID: 0000-0003-0728-0829
[2] Universidad del Cauca (Popayán-Cauca, Colombia). rubendorado@unicauca.edu.co. ORCID: 0000-0002-8537-337X
[3] Ph. D. Universidad del Cauca (Popayán-Cauca, Colombia). cpardo@unicauca.edu.co. ORCID: 0000-0002-6907-2905
[4] Ph. D. Universidad del Cauca (Popayán-Cauca, Colombia). carlosorozco@unicauca.edu.co. ORCID: 0000-0003-2615-3294

needs of the client clearly and accurately represent what they expect. A clear and structured requirement specification process avoids reprocessing at later stages of the project life cycle, generating a benefit in terms of time estimation for new tasks, cost, and effort. In this sense, it is important to have mechanisms or techniques to identify and mitigate possible errors during the requirements specification. Software engineering proposes the term “smell”, which can be defined as a specific symptom that can generate defects in a requirement. The objective of this paper is to establish a broader state of knowledge on the smell identification and classification present during the requirements specification and their impact on the generation of a phenomenon known as requirements debt. This article presents the results obtained after carrying out a systematic mapping of the literature, describing the proposals, initiatives, results, technological tools, benefits and challenges of smell identification and management in the requirements-gathering stage during the software development solutions.

Keywords: requirements debt; requirements engineering; requirements smells; software development; software engineering.

Mapeo sistemático de la literatura sobre los malos olores en los requisitos de desarrollo de software

Resumen

Una de las actividades responsables del éxito en los proyectos de desarrollo de software es la especificación de requisitos, cuyo propósito es asegurar que los deseos o necesidades del cliente representan de forma precisa lo que ellos esperan. Un proceso claro y estructurado durante la especificación de requisitos permite evitar reprocesos en etapas posteriores del ciclo de vida del proyecto, generando un beneficio en términos de estimación de tiempos para nuevas tareas, costo y esfuerzo. En este sentido, es importante contar con mecanismos o técnicas que permitan identificar y mitigar posibles errores durante la especificación de requisitos. En particular, la ingeniería de software propone el término “olor”, que se puede definir como un síntoma concreto que puede generar defectos en un requisito. Con el objetivo de establecer un estado del conocimiento más amplio en torno a la

identificación, clasificación de olores presentes durante la especificación de requisitos y su impacto en la generación de un fenómeno conocido como deuda de requisitos, este artículo presenta los resultados obtenidos después de realizar un mapeo sistemático de la literatura, en el cual se describen las propuestas, iniciativas, resultados, herramientas tecnológicas, beneficios y desafíos en torno a la identificación y gestión de olores en la etapa de levantamiento de requisitos durante el desarrollo de soluciones software.

Palabras clave: desarrollo de software; deuda de requisitos; ingeniería de requisitos; ingeniería de software; olor de requisito.

Mapeamento sistemático de literatura sobre mau cheiro em requisitos de desenvolvimento de software

Resumo

Uma das atividades responsáveis pelo sucesso dos projetos de desenvolvimento de software é a especificação de requisitos, cujo objetivo é garantir que os desejos ou necessidades do cliente representem exatamente o que ele espera. Um processo claro e estruturado durante a especificação de requisitos permite evitar retrabalho em fases posteriores do ciclo de vida do projeto, gerando um benefício em termos de estimativa de tempo para novas tarefas, custo e esforço. Nesse sentido, é importante dispor de mecanismos ou técnicas que permitam identificar e mitigar possíveis erros durante a especificação de requisitos. Em particular, a engenharia de software propõe o termo “cheiro”, que pode ser definido como um sintoma específico que pode gerar defeitos em um requisito. Com o objetivo de estabelecer um estado mais amplo de conhecimento sobre a identificação e classificação de odores presentes durante a especificação de requisitos e seu impacto na geração de um fenômeno conhecido como dívida de requisitos, este artigo apresenta os resultados obtidos após a realização de um mapeamento sistemático do literatura, que descreve as propostas, iniciativas, resultados, ferramentas tecnológicas, benefícios e desafios em torno da identificação e gestão de odores na etapa de levantamento de requisitos durante o desenvolvimento de soluções de software.

Palavras-chave: cheiro necessário; desenvolvimento de software; dívida de requisitos; engenharia de requisitos; engenharia de software.

I. INTRODUCTION

One of the most important processes during the software projects life cycle is requirements engineering [1], whose objective is to define the business needs clearly and precisely and to translate customer needs into tasks that can be implemented at later stages during the solution development process [2]. Requirements engineering is crucial for the success of a software development project since it allows for avoiding reprocesses and cost overruns caused by aspects such as defects caused by ill-defined requirements and additional efforts that arise as a result of little or no requirements management during project execution. In this sense, requirements engineering equips projects with a set of tools to ensure the quality of the requirements [3]. In general, the quality assessment of requirements is performed following the guidelines proposed by the ISO/IEC/IEEE 29148 standard, which proposes that the conformity of a requirement should be performed by systematically identifying a concept known as “smell” [1].

A smell in the context of the requirements is defined as a quality violation, which can lead to a defect with a specific location and detection mechanism [4]. ISO/IEC/IEEE 29148 defines a set of bad smells, including: subjective language, ambiguous adverbs and adjectives, loopholes, open-ended, non-verifiable terms, superlatives, comparatives phrases, negative statements, vague pronouns, incomplete references, among others [4]. Currently, requirements are written in natural language; therefore, quality control on a requirement is carried out by peer reviews [3].

Identifying these smells early in the requirements development process can help detect and correct quality defects before they have a major impact on projects. On the other hand, poor management of smells present in the requirements results in a phenomenon known as Requirements Debt, which can be defined as the distance between the optimal requirements specification and the actual system implementation, under domain assumptions and constraints [19], negatively affecting projects, generating cost overruns, rework and additional efforts as a result of mitigating or solving subsequent defects.

With the objective of obtaining an updated state of knowledge on the proposals, studies and initiatives on the smells identification and classification in the requirements, and how these can cause or avoid the requirements debt, this article presents the results after carrying out a systematic mapping of the literature, in which the initiatives and proposals on this subject were documented and analyzed. The systematic mapping is structured as follows: Section II presents the research method used for the elaboration of the mapping, as well as the execution of the information research. Section III presents the results obtained in response to the research questions. Finally, Section IV presents the conclusions and future work, as well as the discussion of the results and the main observations.

II. METHOD

A systematic mapping of the literature -hereinafter SML- is a process that allows the collecting, categorizing, and structuring of the existing information on a topic of research interest, mainly in the area of Software Engineering [5]. The mapping presented in this paper follows the protocol proposed by Petersen et al. [5] [6], which describes the guidelines for conducting a systematic mapping in Software Engineering. In addition, the guidelines proposed by Kitchenham [7] and Budgen et al. [8] were followed for the research protocol, which is made up of the following stages: (i) definition of research questions; (ii) conduct search for primary studies; (iii) screening of primary studies for inclusion and exclusion criteria; (iv) quality assessment of the primary studies; and (v) data extraction. Figure 1 shows the relationship between the stages performed in the systematic mapping.

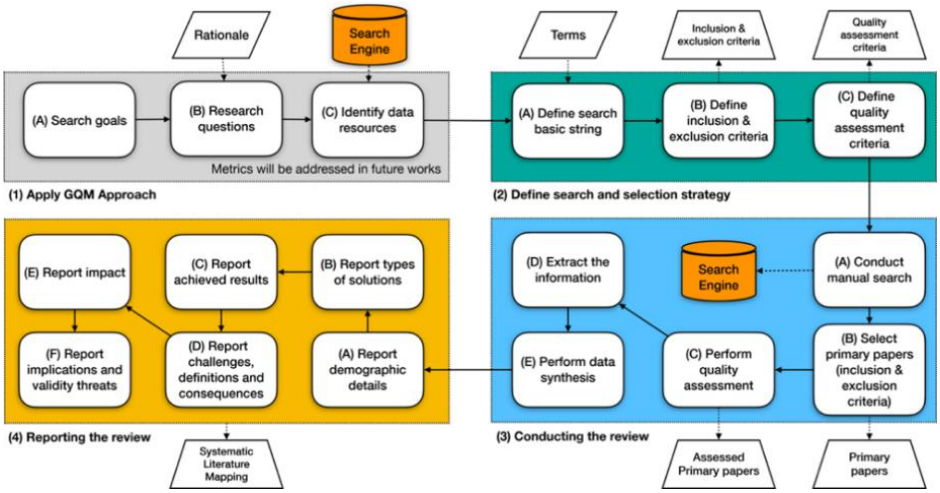

Fig. 1. Stages of the systematic mapping process.

A. Definition of Research Questions

To conduct the SML, a total of three (3) research questions -hereinafter RQs- were defined and presented in Table 1. The RQs categorize the information identified on smells in software development and allow the identification of existing loopholes at the research level.

Table 1. Research Questions.

ID	Research Question	Incentive (Purposes)
RQ1	What kinds of solutions have been proposed?	To know the contribution in one or more of the following categories: (i) conceptual definition, (ii) causes, effects, impacts, and limitations, (iii) evaluation techniques, (iv) technological tools, (v) validation in the industry, (vi) documentation methodologies, (vii) others.
RQ2	Which results have been achieved with the proposals made?	To identify the impact of the proposals made based on the results obtained during their validation in the software industry.
RQ3	Which benefits and challenges does research on the subject entail?	To determine the benefits and challenges for companies to detect and reduce requirements smells associated with the software development life cycle.

B. Conduct Search for Primary Studies

For the search of studies, combinations were made between the keywords identified from a previous review on smells and requirements debt in software development using the logical operators “**AND**” and “**OR**”. As a result, the following basic search string was obtained: (“Requirement Smell” **OR** “Requirement Smells” **OR**

"Requirements Smells") **OR** "Requirements debt" **AND** ("Software development" **OR** "Software engineering" **OR** "requirements engineering"). The string was adapted and applied in seven (7) scientific databases: IEEE Xplore, Science Direct, Scopus, Google Scholar, Springer Link, ACM, and Web of Science (WoS).

C. Screening of Primary Studies for Inclusion and Exclusion Criteria

The selection of relevant studies was carried out at three levels: (i) review of the title, (ii) review of the abstract, introduction, and conclusions, and (iii) review of the full text to determine whether the study met all the inclusion criteria (IC) described in *Table 2*. Subsequently, for screening primary studies, studies that met at least one of the exclusion criteria (EC) described in *Table 3* were discarded.

Table 2. Inclusion Criteria.

ID	Inclusion Criteria (IC)
IC1	Articles whose focus is bad smells in software development requirements.
IC2	Articles whose main subject is the quality requirements in software development.
IC3	Articles whose subject is related to requirements engineering.
IC4	Articles published in journals, prestigious congresses or conferences with peer review.

Table 3. Exclusion Criteria.

ID	Exclusion Criteria (EC)
EC1	Duplicate articles (considering only the most complete and recent that can be evidenced).
EC2	Articles where the research topic is superficially addressed.
EC3	Articles not related to requirements debt during software development.
EC4	Articles of debate type or available only in presentation form or abstracts.
EC5	Articles that are books or book chapters.

D. Quality Assessment of Primary Studies

In addition, the quality of the primary studies was assessed to determine their possible relevance in the future. The assessment was based on the instrument proposed by Kitchenham [11] and an adaptation of the assessment system proposed in [12]. As a result, a questionnaire of eleven (11) criteria was constructed and organized into 5 categories: clarity, quality, credibility, relevance, and rigor. To evaluate the criteria, a three-value scoring system (-1, 0, +1) was defined, which is presented in detail at https://tinyurl.com/29bs5lgs. It is important to clarify that the

score obtained by an article is not considered an exclusion criterion; the score obtained is used to know the relevance that an article could have in the future.

E. Data Extraction

The extraction of relevant information from each study was carried out by defining a template that presents elements such as problem addressed, type and methodology of research, type and proposed solution, among others. The template made it possible to standardize and facilitate the extraction of relevant information from each study. The template can be consulted at https://tinyurl.com/2chsxq5u.

III. RESULTS

In total, seven (7) iterations were performed, one for each database. Since each database has its own configuration, it was necessary to adapt the search string regarding the original string (the adapted strings can be consulted at https://tinyurl.com/27btxspz). According to the results presented in Table 4, 533 studies were identified, of which 132 relevant studies were initially selected. After a detailed review, 46 repeated relevant studies were eliminated, resulting in a total of 86 relevant studies after applying the ICs. Subsequently, the ECs were applied to eliminate 62 studies. Finally, a total of 24 studies were obtained, which are considered primary studies, hereafter PS.

Table 4. Research results.

No.	Data Source	Identified Studies	Relevant Studies	Repeated Studies	Primary Studies
1	Google Scholar	275	56	0	19
2	Scopus	128	61	31	5
3	Science Direct	24	4	4	0
4	Springer Link	46	2	2	0
5	IEEE Xplore	41	6	6	0
6	ACM	18	2	2	0
7	WoS	1	1	1	0
Total		533	132	46	24

In addition, a backward snowballing [17] of the PSs references was performed. As a result of this review, four (4) additional studies were selected, becoming a total of 28 PSs. Due to space limitations, details of all PSs can be found at

https://tinyurl.com/2cqzcn2p, hereafter, articles are referenced with the acronym A, as shown at https://tinyurl.com/2cqzcn2p. The contribution of each primary study to answering the research questions posed in *Table 1* is described at https://tinyurl.com/25xxj3bg. The following subsections present the results for each research question defined in this systematic mapping.

A. *What Kinds of Solutions Have Been Proposed?*

As shown in Figure 2, 3.6% of the studies (A1) mention a conceptual definition, where a set of smells associated with the requirements are proposed: subjective language, ambiguous adverbs and adjectives, loopholes, open-ended, non-verifiable terms, superlatives, comparatives phrases, negative statements, vague pronouns and incomplete references. On the other hand, 3.6% of the studies (A13) are described in the category of causes, effects, impacts or limitations. 28.6% of the studies (A3, A7, A11, A12, A14, A16, A17, A19) focus on the definition of assessment methods or techniques. For example, in A3, ISO/IEC/IEEE 29148 [43] and IEEE-830-1998 [44] standards were analyzed to assess whether early smell detection and correction can help improve quality reviews. In A7, 870 requirements were analyzed and categorized, detecting how many of them have requirements smells and how these are related to the use of subjective language, incomplete references or non-verifiable terms, analyzing their results in a set of examples against assessments made by humans, with the purpose of identifying ambiguous terms between different domains and classifying them by ambiguity score. Also, 32.1% of the studies (A2, A5, A15, A20, A21, A23, A24, A25, A28) concentrate on the development of technological tools. For instance, A15 proposes the Quality User Story framework to detect quality defects and suggest possible solutions based on 13 quality criteria for user stories (US). US are well-structured, atomic, minimalist, conceptually sound, problem-oriented rather than solution-oriented, unambiguous, non-conflicting, complete and well-formed sentences, unique, uniform, independent and complete. Furthermore, 21.4% of the studies (A4, A6, A10, A18, A26, A27) concentrate on performing industry validation; for example, in A6, a Natural Language Processing (NLP) tool is used to empirically validate the effect that quality

defects have on test cases designed in later stages of a project. In A18, we study the problems when interpreting semantic relations of the type "If A and B then C" in the wording of the requirements since they represent a source of ambiguity. Finally, 10.7% of the studies (A8, A9, A22) are systematic literature reviews. In A9, we sought to know the criteria that should be considered to evaluate the quality of requirements in the context of agile software development using the criteria proposed by the INVEST mnemonic (Independent, Negotiable, Valuable, Estimable, Small and Testable), proposed by Bill Wake in 2003, which proposes the characteristics that should be considered to ensure the quality of a user story [45]. Nevertheless, our study included additional criteria such as completeness, consistency and uniformity of a user story.

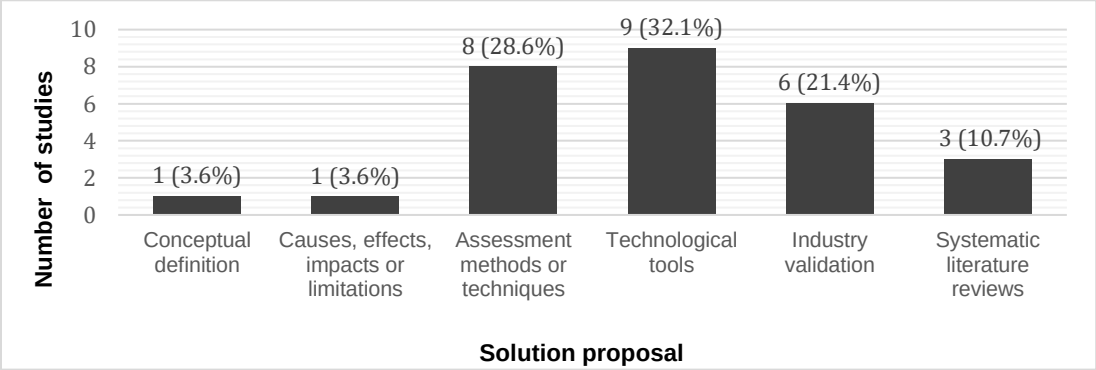

Fig. 2. Number of studies per solution proposal.

B. Which Results Have Been Achieved with the Proposals Made?

As shown at https://tinyurl.com/25rlkcp8, the studies (A1, A2, A3, A5, A6, A8, A9) define bad smells in the requirements as indicators of a quality violation that can lead to a defect. In (A1, A2, A3, A5, A7, A8, A9, A23) they are based on ISO/IEC/IEEE 29148 to describe the following smells: subjective language, ambiguous adverbs and adjectives, loopholes, open-ended terms, superlatives, comparatives phrases, negative statements, vague pronouns and incomplete references. On the other hand, the study (A7) relies on the ISO/IEC 25010 standard to define the following smells: functional suitability, performance efficiency, compatibility, usability, reliability, security, maintainability and portability. In A8, smells are defined as signs

of inaccuracy or ambiguity in the requirements statement. In (A9, A22), some sources of poor requirements quality are defined, as well as incompleteness and ambiguity. In (A22, A25), smells are defined using concepts such as vagueness, language problems and ambiguity. As a result, these studies propose a classification of requirements smells according to their lexical, syntactic, semantic and pragmatic conformity. Finally, only one of the studies proposes a definition related to the concept of requirements debt (A4), describing it as the distance between optimal requirements and the actual system implementation.

According to the analysis of the related works, it was possible to identify 15 smells present in the literature. Some of them are based on the taxonomy of the ISO/IEC/IEEE 29148 standard. Among them are: subjective language, ambiguous adverbs and adjectives, loopholes, open-ended, non-verifiable terms, superlatives, comparatives phrases, negative statements, vague pronouns, incomplete references, among others. Details of each of the smells identified in the SML are available at https://tinyurl.com/2aquzp8y. Furthermore, a total of 10 causes that generate bad smells in the requirements in software development were identified. Some identified causes are: requirements written in natural language or passive voice, limited or non-existent knowledge in the software engineering domain, use of conditionals in defining requirements, informal analysis of requirements, among others. In addition, it was possible to identify a total of 11 critical effects that occur in the presence of bad smells in requirements. Among them are: cost and time overruns in the software development life cycle, reprocessing, misunderstandings between the creators and readers of the requirements documents affecting test cases, problems in processes subsequent to the definition of software requirements, among others (the detail of causes and effects can be consulted at https://tinyurl.com/2de6lftk). Similarly, a total of 5 good practices proposed in the primary studies were observed to prevent bad smells in the requirements and anticipate their effects. These practices are: automatic review of requirements, defining measurable requirements, validation of bad smells in requirements before defining test cases, taking into account the positive and negative cases in automatically generated test cases and preprocessing requirements written in

natural language before moving to the design phase. In general, all practices focus on the prevention of bad smells in requirements in software development. Regarding the use of tools, the studies mention different implementations for smell detection and quality assessment in the requirements (A1, A15, A23, A25, A26, A27, A28). Details of these practices and tools are available at https://tinyurl.com/263jynnb.

Finally, A11 proposes a Natural Language Processing approach to identify ambiguous terms across different domains and rank them by ambiguity score. A16 proposes a series of steps to ensure the quality of the requirements. A22 shows a state-of-the-art review of different NLP-based strategies for ambiguity detection, and A19 presents a set of requirements data to use in NLP tool development processes.

C. What are the Benefits and Challenges of Researching this Topic?

According to the results presented in A2, A3, A6, A24 and A25, carrying out a thorough investigation on smell identification in the requirements early in the software development process reduces the incidence of defects in later stages of development, saving money, effort and time. On the other hand, test cases can be generated early (A6) and project estimates are more accurate (A9). In addition, the use of approaches or techniques that apply automatic natural language processing allows mitigating human limitations when detecting quality defects in the requirements (A24). Also, using NLP approaches speeds up the review of quality defects, e.g., the tool defined in A25 detects four times more genuine ambiguities than an average human analyst.

In relation to the challenges, the possibility of developing new smell detection techniques in requirements has been raised to increase the understanding of which defects can be revealed by these smells and which cannot (A1, A3). In A13, six (6) challenges were identified when using NLP systems for requirements analysis, including: (i) improvement of coreference resolution, (ii) extension of the scope of inputs for NLP systems, visual, auditory, among others, (iii) reduction of ambiguity, (iv) support to the use of domain ontologies, (v) improvement of the accuracy of natural language processing in requirements, and (vi) improvement of algorithms for detecting ambiguity. A5 concludes the need to extend the existing requirements data

sets for natural language processing in multiple applied studies. A4 and A11 make explicit the need to establish strategies to improve the detection of ambiguities according to the business domain in the requirements gathering and analysis phases. Additionally, challenges were identified related to the need of studying different techniques for smell detection using criteria such as: subjective language, comparative phrases, superlatives, among others, and to know their impact on the quality of the artifacts resulting from the requirements specification. Finally, the primary studies are based on the detection of smells written in English, which limits the study of smell identification or classification in other languages.

IV. Discussion and Conclusions

Results were presented at the SML that reveal the current state of research related to requirements smells and their consequences in the software development life cycle. According to the analysis of the results, the studies focus on smell identification and classification, but it was not possible to observe their extended application in exhaustive case studies. In this sense, further application in real-world environments is required to enable future software development projects to understand, manage and improve how they can detect and mitigate existing smells during the requirements specification stage. Furthermore, the SML allowed us to identify multiple definitions of the term "smell" in requirements engineering and define smells (in general) as concrete symptoms of quality defects in the requirements specifications. In addition, fifteen (15) requirements smells (https://tinyurl.com/2aquzp8y), ten (10) causes (https://tinyurl.com/2de6lftk), eleven (11) effects (https://tinyurl.com/2de6lftk) and five (5) good practices (https://tinyurl.com/263jynnb) were identified. Similarly, different methods and software tools are proposed as a solution to automate the smell identification process in the requirements (https://tinyurl.com/263jynnb). The tools found concentrate on the identification of the smells defined by the ISO/IEC/IEEE 29148 standard (https://tinyurl.com/23x4j8so) and implement detection mechanisms for each of them (https://tinyurl.com/2aquzp8y); among the most common are: dictionaries, morphological analysis and part-of-speech (POS) tagging. In addition,

proposals were observed to automate the detection of bad smells in the requirements. However, there was no evidence of studies that support the detection of bad smells in projects where the requirements specification is in languages other than English.

Finally, with the development of this work, different opportunities for future work can be identified, as well as: (i) to broaden the conceptual definition of smells in requirements in software development and the usefulness that they can have for quality control, (ii) to establish new techniques or models capable of measuring the level of uncertainty in the quality artifacts used during the requirements specification phase of software development, and (iii) to improve natural language processing techniques, searching for solutions to the limitations and challenges that are currently present in NLP models.

AUTHORS' CONTRIBUTION

Mayra-Alejandra Castillo-Motta: Research, data analysis, model definition, implementation, validation.

Rubén-Darío Dorado-Córdoba: Research, data analysis, model definition, implementation, validation.

César-Jesús Pardo-Calvache: Research, Supervision, Methodology, Validation, Writing – review and editing.

Carlos-Eduardo Orozco-Garcéz: Research, Supervision, Methodology, Validation, Writing – review and editing.

REFERENCES

[1] M. K. Habib, S. Wagner, D. Graziotin, "Detecting Requirements Smells with Deep Learning: Experiences, Challenges and Future Work," *IEEE International Conference on Requirements Engineering*, vol. 2021, pp. 153–156, 2021. https://doi.org/10.1109/REW53955.2021.00027

[2] Junta de Andalucía, *Ingeniería de requisitos | Marco de Desarrollo de la Junta de Andalucía*, 2021. https://www.juntadeandalucia.es/servicios/madeja/contenido/subsistemas/ingenieria/ingenieria-requisitos

[3] H. Femmer, "Reviewing Natural Language Requirements with Requirements Smells-A Research Proposal," in *11th International Doctoral Symposium on Empirical Software Engineering*, 2013.

[4] H. Femmer, D. Méndez Fernández, S. Wagner, S. Eder, "Rapid Quality Assurance with Requirements Smells," *Journal of Systems and Software*, vol. 123, pp. 190-213. https://doi.org/10.1016/j.jss.2016.02.047

[5] K. Petersen, S. Vakkalanka, L. Kuzniarz, “Guidelines for conducting systematic mapping studies in software engineering: An update,” *Information and Software Technology*, vol. 64, pp. 1–18, 2015. https://doi.org/10.1016/j.infsof.2015.03.007

[6] K. Petersen, R. Feldt, S. Mujtaba, M. Mattsson, “Systematic Mapping Studies in Software,” *International Journal of Software Engineering & Knowledge Engineering*, vol. 17, no. 1, pp. 33–55, 2007.

[7] B. Kitchenham, S. Charters, “Guidelines for performing Systematic Literature Reviews in Software Engineering, Version 2.3,” *EBSE Technical Report, Keele University,* 2007. https://doi.org/10.1145/1134285.1134500

[8] D. Budgen, M. Turner, P. Brereton, B. Kitchenham, “Using Mapping Studies in Software Engineering,” *Lecture Notes in Business Information Processing*, vol. 32, pp. 441–442, 2009. https://doi.org/10.1007/978-3-642-02152-7_36

[9] V. R. Basili, G. Caldiera, “The Goal Question Metric Paradigm,” *Encyclopedia of Software Engineering*, vol. 2, pp. 528–532, 2000.

[10] R. Wieringa, N. Maiden, N. Mead, C. Rolland, “Requirements engineering paper classification and evaluation criteria: A proposal and a discussion,” *Requirements Engineering*, vol. 11, no. 1, pp. 102–107, 2006. https://doi.org/10.1007/s00766-005-0021-6

[11] Anonym, *State of Agile Report | State of Agile*, 2020. https://digital.ai/resource-center/analyst-reports/state-of-agile-report#ufh-c-7027494-state-of-agile

[12] T. Dybå, T. Dingsøyr, “Strength of Evidence in Systematic Reviews in Software Engineering,” in *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*, 2008, pp. 178–187. https://doi.org/10.1145/1414004.1414034

[13] L. Yang *et al.*, “Quality Assessment in Systematic Literature Reviews: A Software Engineering Perspective,” *Information and Software Technology*, vol. 130, e106397, 2021. https://doi.org/10.1016/j.infsof.2020.106397

[14] M. Ivarsson, T. Gorschek, “A method for evaluating rigor and industrial relevance of technology evaluations,” *Empirical Software Engineering*, vol. 16, no. 3, pp. 365–395, 2011. https://doi.org/10.1007/S10664-010-9146-4/figures/5

[15] SCImago, *Scimago Journal & Country Rank*, 2022. https://www.scimagojr.com/

[16] CORE Inc, *Core Conference Ranks*, 2022. http://portal.core.edu.au/conf-ranks

[17] S. Jalali, C. Wohlin, “Systematic literature studies: Database searches vs. backward snowballing,” *International Symposium on Empirical Software Engineering and Measurement*, 2012, pp. 29–38. https://doi.org/10.1145/2372251.2372257

[18] H. Femmer, D. M. Fernández, E. Juergens, M. Klose, I. Zimmer, J. Zimmer, “Rapid requirements checks with requirements smells: Two case studies,” in *1st International Workshop on Rapid Continuous Software Engineering*, 2014, pp. 10–19. https://doi.org/10.1145/2593812.2593817

[19] V. Lenarduzzi, D. Fucci, D. Mendéz, “On the Perceived Harmfulness of Requirement Smells: An Empirical Study,” in *26th International Working Conference on Requirement Engineering: Foundation for Software Quality*, 2020.

[20] A. Beer, M. Junker, H. Femmer, M. Felderer, “Initial investigations on the influence of requirement smells on test-case design,” in *IEEE 25th International Requirements Engineering Conference Workshops,* 2017, pp. 323–326. https://doi.org/10.1109/REW.2017.43

[21] A. T. Seidel Calazans *et al.*, "Quality requirements and the requirements quality: The indications from Requirements Smells in a Financial Institution Systems," in *ACM International Conference Proceeding Series*, 2019, pp. 472–480. https://doi.org/10.1145/3350768.3350782

[22] R. Nascimento, E. Aranha, U. Kulesza, M. Lucena, "Requirements Smells as indicators of poor quality in requirement specification: A systematic mapping of literature," in *21st Workshop em Engenharia de Requisitos,* 2018.

[23] R. Nascimento, E. Guimarães, M. Lucena, "Requirements Smells como Indicador de Qualidade para Histórias de Usuários: Estudo Exploratório," in *WER*, 2021.

[24] D. Šenkýř, P. Kroha, "Problem of incompleteness in textual requirements specification," in *Proceedings of the 14th International Conference on Software Technologies*, 2019, pp. 323–330.

[25] A. Ferrari, A. Esuli, "An NLP approach for cross-domain ambiguity detection in requirements engineering," *Automated Software Engineering*, vol. 26, no. 3, pp. 559–598, 2019. https://doi.org/10.1007/S10515-019-00261-7

[26] H. Femmer, J. Mund, D. M. Fernandez, "It's the Activities, Stupid! A New Perspective on RE Quality," in *2nd International Workshop on Requirements Engineering and Testing,* 2015, pp. 13–19. https://doi.org/10.1109/RET.2015.11

[27] A. Alzayed, A. Al-Hunaiyyan, "A Bird's Eye View of Natural Language Processing and Requirements Engineering," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 5, pp. 81–90, 2021.

[28] M. Unterkalmsteiner, T. Gorschek, "Requirements Quality Assurance in Industry: Why, What and How?," *Lecture Notes in Computer Science*, vol. 10153, pp. 77–84, 2017. https://doi.org/10.1007/978-3-319-54045-0_6

[29] L. Garm, F. Dalpiaz, J. Van Der Werf, S. Brinkkemper, "Improving agile requirements: the Quality User Story framework and tool," *Requireriments Engineering*, vol. 21, no. 3, pp. 383–403, 2016. https://doi.org/10.1007/s00766-016-0250-x

[30] H. Femmer, M. Unterkalmsteiner, T. Gorschek, "Which requirements artifact quality defects are automatically detectable? A case study," in *IEEE 25th International Requirements Engineering Conference Workshops,* 2017, pp. 400–406. https://doi.org/10.1109/REW.2017.18

[31] Y. Al-Kasabera, W. Alzyadat, A. Alhroob, S. al Showarah, A. Thunibat, "An automated approach to validate requirements specification," *International Journal of Advanced Computer Technology*, vol. 9, no. 2, pp. 3578–3585, 2020.

[32] J. Fischbach, J. Frattini, D. Mendez, M. Unterkalmsteiner, H. Femmer, A. Vogelsang, "How Do Practitioners Interpret Conditionals in Requirements?," in *22nd International Conference on Product-Focused Software Process Improvement,* 2021. https://doi.org/10.1007/978-3-030-91452-3_6

[33] A. Ferrari, S. Gnesi, G. O. Spagnolo, "PURE: A Dataset of Public Requirements Documents," in *IEEE 25th International Requirements Engineering Conference*, 2017, pp. 502-505. https://doi.org/10.1109/RE.2017.29

[34] E. Sarmiento-Calisaya, E. H. Cárdenas, V. Cornejo-Aparicio, G. S. Alzamora, "Towards the improvement of natural language requirements descriptions: The C&&L tool," in *Proceedings of the ACM Symposium on Applied Computing*, 2020, pp. 1405–1413. https://doi.org/10.1145/3341105.3374028

[35] A. Rani, G. Aggarwal, “Algorithm for automatic detection of ambiguities from software requirements,” *International Journal of Innovative Technology and Exploring Engineering*, vol. 8, no. 9, pp. 878–882, 2019. https://doi.org/10.35940/ijitee.I1141.0789S19

[36] M. Q. Riaz, W. H. Butt, S. Rehman, “Automatic Detection of Ambiguous Software Requirements: An Insight,” in *5th International Conference on Information Management,* 2019, pp. 1–6. https://doi.org/10.1109/INFOMAN.2019.8714682

[37] M. H. Osman, M. F. Zahrin, “Ambi Detect: An Ambiguous Software Requirements Specification Detection Tool,” *Turkish Journal of Computer and Mathematics Education*, vol. 12, no. 3, pp. 2023–2028, 2021. https://doi.org/10.17762/TURCOMAT.V12I3.1066

[38] A. O. J. Sabriye, W. M. N. Wan Zainon, “An approach for detecting syntax and syntactic ambiguity in software requirement specification,” *Journal of Theoretical and Applied Information Technology*, vol. 96, no. 8, pp. 2275–2284, 2018.

[39] B. Gleich, O. Creighton, L. Kof, “Ambiguity detection: Towards a tool explaining ambiguity sources,” *Lecture Notes in Computer Science*, vol. 6182, pp. 218–232, 2010. https://doi.org/10.1007/978-3-642-14192-8_20/

[40] E. Juergens *et al.*, “Can clone detection support quality assessments of requirements specifications?,” in *International Conference on Software Engineering*, 2010, pp. 79–88. https://doi.org/10.1145/1810295.1810308

[41] V. Antinyan, M. Staron, “Rendex: A method for automated reviews of textual requirements,” *Journal of Systems and Software*, vol. 131, pp. 63–77, 2017. https://doi.org/10.1016/J.JSS.2017.05.079

[42] N. Carlson, P. Laplante, “The NASA automated requirements measurement tool: a reconstruction,” *Innovations in Systems and Software Engineering*, vol. 10, no. 2, pp. 77–91, 2013. https://doi.org/10.1007/S11334-013-0225-8

[43] ISO, *ISO - ISO/IEC/IEEE 29148:2011 - Systems and software engineering — Life cycle processes — Requirements engineering*, 2011. https://www.iso.org/standard/45171.html

[44] G. Mendez, *Especificación de Requisitos según el estándar de IEEE 830*, 2008. https://www.fdi.ucm.es/profesor/gmendez/docs/is0809/ieee830.pdf

[45] W. Bill, *INVEST in Good Stories, and SMART Tasks - XP123*, 2003. https://xp123.com/articles/invest-in-good-stories-and-smart-tasks/