BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **24**, No 4
Sofia • 2024 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2024-0037

Software Requirement Smells and Detection Techniques: A Systematic Literature Review

Esubalew Alemneh[1], *Fekerte Berhanu*[2]

[1]*ICT4D Research Center, Bahir Dar Institute of Technology, Bahir Dar University, P.O.Box 26, Bahir Dar, Ethiopia*
[2]*Department of Software Engineering, Kombolcha Institute of Technology, P.O.Box, Kombolcha, Ethiopia*
E-mails: esubalew.alemneh@bdu.edu.et fekerlight@gmail.com
ORCIDs: https://orcid.org/0000-0002-1970-3537 https://orcid.org/0009-0006-2304-3443

Abstract: *One of the major reasons for software project failure is poor requirements, so numerous requirement smells detection solutions are proposed. Critical appraisal of the proposed requirement fault detection methods is crucial for refining knowledge of requirement smells and developing new research ideas. The objective of this paper was to systematically review studies that focused on detecting requirement discrepancies in textual requirements. After applying inclusion and exclusion criteria and forward and backward snowball sampling techniques using database-specific search queries, 19 primary studies were selected. A deep analysis of the studies shows that classical NLP-based requirement smells detection techniques are the most commonly used ones and ambiguity is the requirement smell that has the utmost attention. Further investigation depicts the scarcity of open-access datasets, and tools employed to detect requirement faults. The review has also revealed there is no comprehensive definition and classification of requirement smells.*

Keywords: *Requirements, Software requirement specification, Requirement smells, Ambiguity, Requirement smell detection, Natural language processing.*

1. Introduction

Requirement engineering is the process of eliciting, analyzing, specifying, validating, and managing requirements of large and complex software systems, and as such it comprises critical activities in software engineering [1, 2]. It is an initial stage of the Software Development Life Cycle (SDLC) to create high-quality software requirements, which eventually should result in high-quality and cost-effective software delivered in time. It provides an appropriate mechanism to understand and analyze customer needs, specify and validate the specifications, and manage the requirements. Software requirements are a description of what the system should do (functional requirements) and the qualities the system should possess while offering the functions (non-functional requirements) [2-5] The description of the intended purpose and environment of software under development is presented in one of the significant outputs of the software development process which is Software Requirements Specification (SRS) document. SRS is a specification and elucidation of software requirements that are supposed to satisfy customer needs and increase product quality.

During software development, the quality of specified requirements has a direct impact on the success and failure of the software development project [6-8]. Therefore, software requirements should assuage the characteristics of individual requirements as well as requirements together in an SRS document [9, 10]. Developing qualified software requires well-written software requirements with no indicator of defects as the quality of requirements has a great impact on the success of the software project [2, 4, 10]. Hence, to produce quality software, problems with requirements must be detected at the earliest stage of SDLC and removed without further ado. The later the requirement errors are identified, the more difficult and expensive it becomes to correct them [10]. For instance, 11] found that requirement defects are 10 to 200 times more expensive to fix once they have infiltrated the system compared to being detected at the initial stage of development. In large-scale software development projects, requirement problems gradually evolve to design, code and test case smells, causing unprecedented risk in software development projects. Indicators of software requirement defects or quality problems in software requirements are known as requirement bad smells or just requirement smells [10]. Since, poor quality requirements are reasons for software project failure, ensuring requirement quality is an important activity in the software development process [11]. Despite initiatives and efforts to write quality requirements and SRS documents, the software industry is dodged with poor requirements. Literature has shown that 40% to 60% of smells in software development phases are because of poor quality requirements [3, 12].

Several studies have been conducted on identifying requirement smells in both textual requirements and requirements written in other forms. To detect bad smells from the use case description S e k i, H a y a s h i and S a e k i [13], proposed an automated bad smell detection approach. A way to detect requirement smells from linguistic goal models is proposed in [14] and the impact of requirement smells on test case design is studied in [15]. Various techniques ranging from traditional dictionary-based detection to advanced machine learning techniques [16] have been applied for smell detection. For instance, a multi-class multi-label requirements smell classification is proposed using deep learning [17]. Besides studies that aim at the analysis and improvement of requirement quality, numerous studies from academia and industry are devoted to detecting and classifying requirement smells and assessing their impact.

In this study, we appraise the outputs of the research on requirement smells and detection approaches via Systematic Literature Review (SLR). The SLR is vital to refine the knowledge on requirement smell, to develop new research ideas, to gain critical skills in synthesizing existing literature, and to combine findings from different studies and discover new findings. Even though plenty of SLRs are available on code smells [18, 19], and test smells [20], as far as our knowledge there is no SLR on requirement smells. However, a systematic mapping study was conducted in [21] to create and structure research areas in relation to requirement smells. The study provided an overview of requirement smells in general terms. Moreover, in [7] a classical literature review on analyzing the quality of textual requirements using Natural Language Processing (NLP) was conducted. However, the focus of the review was on tools to analyze requirements using quality models.

This systematic literature review aimed to identify different kinds of requirement smells and to assess the existing smell detection techniques and tools on textual requirements using guidelines formulated by Kitchenham and Charters [22] and PRISMA reporting guidelines for software engineering [23]. The SLR is registered at OSF REGISTRIES and its Peer Review of Electronic Search Strategies (PRESS) with registration (DOI: 10.17605/OSF.IO/VZ83P).

The rest of the paper is organized as follows. Section two explains the background on requirement smells and techniques to detect the bad smells. Next, the research methodology is presented in detail. Results and discussions are elucidated in section four. Threats in the validity and future research direction are discussed in section five and six, respectively. Finally, conclusions are drawn in section seven.

2. Background

2.1. Requirement smells

In software engineering, "bad smell" refers to symptoms that show something is wrong with an artifact produced during the software development process or bad practices in developing software [24]. Most literature attributes bad smell in software engineering to computer programs [18, 25-27]. However, there are bad smells on requirements [4], on architecture [28], on design [29], and on testing [20, 30]. In some literature, bad smells together are termed software smells [31, 32]. Bad smells are also viewed from another dimension as usability smells [33], services antipatterns/smells [34], aspect-oriented system smells [35], object-oriented system smells [25], software product line smells [36], configuration smells [26], etc. However, even though bad smells can be classified based on different angles, the smells can be traced back to problems in requirements, architecture, design, code, or tests. All types of bad smells must be detected and removed with no exception.

Definitions of a software "smell" provided at [24] don't address the requirement small. The phrase "Requirement smell" was originally coined by [10] in 2013 and it originates from code smell which was introduced by Fowler and Beck [27]. Since the introduction of requirement smells, various definitions have been provided for requirement smells by different authors from their perspectives. Table 1 provides a consolidated list of definitions for requirement smells. A comprehensive and agreeable definition for requirement smells is yet to come from scholars as the existing definitions exclude some aspects related to ill-defined requirements. For instance, [8] associates requirement smell to functional requirements only whereas [37] states that only quality indicators that are evolved to defects are termed requirement smells.

Table 1. Definitions of requirement smells

Definition of requirement smell	Reference
Requirement smell is a concrete symptom of a requirement artifact’s quality defect in the usage context of a certain activity	[10]
Requirement smells are indicators of a quality violation, which may lead to a defect, with a concrete location and a detection mechanism	[37]
Requirement smells are quality problems in functional requirements that can lead to defects at different levels of severity	[8]
Requirement smells or feature request smells are quality defects on requests/ requirements	[38]
Requirement smells are concrete symptoms of a requirement artifact’s quality defect	[4]
Requirement smells are quality violations in requirements or are defects in requirements	[39]
Requirement smells as a specific symptom that can generate defects in a requirement	[21]
Requirement smells are software requirements that suffer from imprecision, ambiguity, and other quality issues	[40]

Generally, requirement smells are indicators of poor quality of software requirements. We can say that a requirement has poor quality if the requirement does not satisfy quality standards set at [3] and [9] or if the requirement doesn’t meet user needs. Quality of requirements may be attributed to individual requirements or a group of requirements altogether as presented in the SRS document (Table 2).

Table 2. Characteristics of individual requirements and set of requirements

Classification by	Individual requirements	Group of requirements in SRS
Katasonov and Sakkinen [41]	Complete, correct, unambiguous, feasible, necessary, prioritized, verifiable, concise	Complete, consistent, non-redundant, traceable, organized, conformant to standards
Femmer [10]	Necessity, implementation freeness, non-ambiguity, completeness, singularity, feasibility, traceability, verifiability	Completeness, consistency, affordability, and boundedness
ISO/IEC/IEEE 29148 [9]	Necessary, appropriate, correct, confirming, unambiguous, complete (requirement level completeness), singular, feasible, verifiable	Complete, consistent, feasible, comprehensible, able to be validated

Requirements may be written using natural language, structured natural language (tables and diagrams), or using formal methods. Despite writing requirements using natural language has proven advantages, natural languages are inherently ambiguous, imprecise, and subject to misinterpretation. Formal language effectively addresses ambiguity, but it is costly and it is not understood by most stakeholders. Hence, most requirements (about 79%) are written in natural languages [42].

Requirement smells can be categorized in a different way. A catalog of bad smells in use case descriptions developed at [43]. Walia and Carver developed a taxonomy of errors that may occur during the requirements phase by referring to 149 papers [44]. The errors are broadly categorized as people errors, process errors, and documentation errors. Requirement quality can also be classified based on the characteristics that interest developers/engineers or clients [45]. Validability, completeness, consistency, and precision are concerns of clients while verifiability, modifiability, understandability, unambiguity, traceability, abstraction, and atomicity are desirable properties of requirements as seen by engineers.

Table 3. Taxonomies of requirement smells

Smell category	Smells
Morphological requirement smells	Very long/short sentences, very long/short paragraphs, unreadability (long words, long and complex sentences), excessive punction, lack of punctuation, excessive use of acronyms, use of abbreviations
Lexical requirement smells	use of copulative-disjunctive terms (alternative terms/phrases = non-atomic requirement, optional, more than one verb, coordinating conjunction, use of coordinator), excessive use of negative terms, too many control flow terms, use of anaphorical terms (alternative terms/phrases = vague pronouns, demonstrative adjective), use of imprecise or subjective terms (alternative terms/phrases = subjective language, ambiguous adverbs and adjectives, loopholes, open-ended, non-verifiable terms, superlatives, comparatives, not precise verb, ambiguity (coordination, referential, lexical, syntactic, semantic, pragmatic, attachment, analytical), vagueness/vague terms and symbols, weakness, generality (not specific), speculative expression, non-explicit requirements (implicit), use of modal adverbs, use of quantifiers), use of design or technology related terms (alternative terms/phrases = not requirement, use of pseudocode and control-flow expression, requirement expressing a solution (design))
Analytical requirement smells	Excessive use of imperative forms, not having at least one imperative verb (imperative), usage of conditional mood or non-assertive requirement (conditional), passive voice, excessive domain terms, alternative terms/phrases (use of a large number of domain concepts too many, domain verbs), rare use of domain terms
Relational requirement smells	Excessive number of versions, too low or too high nesting, excessive number of dependencies, too low or too high coupling of requirements
Incompleteness and Language related requirement smells	Incomplete requirement (alternative terms/phrases = incomplete references, incomplete condition / missing condition, incomplete system response, incorrect order requirement, missing description, partial content, incomplete enumerations, justifications in the requirement (rationale), missing unit of measurement, use of continuance, use of directives), undefined terms, language error, use of plural nouns

Krogstie and Lindland [46] proposed a requirement quality framework that has four parts: syntactic quality (correctness of the language used to express the requirement), semantic quality (validity and completeness of the requirements), pragmatic quality (comprehensibility of the requirements by audience), and social quality (level of agreement between stakeholders). However, the latter three groups of quality are immeasurable in practice and evaluation can be made only heuristically. Authors of [44] adapted the classification from [45] and organized textual requirements into four groups based on requirement quality indicators. Requirement smells can be classified into four categories which correspond to the four requirement quality indicators as requirement smells are indicators of quality violation of requirements [37]. Any requirement smell that doesn’t fit into the four categories could be included in the fifth category which we named *incompleteness and language-related requirement smells* (Table 3). We used the definitions and/or descriptions provided at selected papers, to allocate the individual smells to the smell categories. The classification is open to debate as few smells have overlapping definitions, and some others conflict with each other.

2.2. Smell detection methods

The primary issue with requirements quality is that they are written in natural language, which lacks formal semantics. This makes them difficult to understand and prone to ambiguity and vagueness [4, 25]. Moreover, the processes and tools used in the requirement engineering process can affect the quality of requirements [47, 48]. Additionally, the majority of requirements originate from stakeholders who have limited domain knowledge and have often conflicting needs [14, 18]. Customers don't tell all the requirements at the outset, rather they come with new requirements after implementation. The problem may also come from the requirement engineer due to a lack of experience in writing requirements or understanding customers' needs. Requirement smells need to be detected as early as possible to minimize their impact in subsequent phases of SDLC [21, 49] especially when the iteration is long and feedback comes late [4]. This is because flaws in requirements cause software project delays, reworks, and low customer satisfaction [40, 50]. Therefore, early detection and removal of smells from requirement statements play its parts to produce quality design, code, and test cases. Literature has shown that more than half of the errors detected during software development happen in the requirement specification stage [25, 36]. That is why the identification of requirement smells has a vital role in the success of software projects by ensuring the quality of requirements [8, 27, 40, 51].

However, automatic requirement smell detection is very challenging in comparison with smell detection at the coding and design phases as the latter involves formal representations [35]. Hence, many requirements for smell detection approaches depend on manual reviews at various levels of formality (informal, formal, walkthrough, inspection, audit) [52]. However, it is not feasible to manually review SRS documents of large and complex systems [4, 53]. Therefore, manual review of SRS needs to be supported by automatic smell detection CASE (Computer Assisted Software Engineering) tools. Various tools that employ different smell detection techniques have been already proposed. Some requirement smell detections rely on Natural Language Processing (NLP) [8, 37, 38, 54, 55] NLP-based techniques involve analysis of the syntactic (arrangement of words) and the semantic (meaning) characteristics of requirements written in natural languages. NLP techniques such as the use of language models, Part-Of-Speech (POS) tagging, tokenization, lemmatization, etc. have been reported in the literature.

Rule-based requirement smell detection technique is the other approach reported in some papers [56-59]. In this method, rules are defined, usually manually, by experts to detect one or more required bad smells. Then, the requirements or SRS documents are evaluated against the rules. The other method is the knowledge dictionary-based requirement smell identification technique [60]. In this method, a dictionary is constructed manually from some language elements (e.g., from transitive verbs and their associated object words [60]) and used for smell detection. A huge effort due to extensive human involvement in the rule and knowledge-based methods has forced scholars to look for other smell detection mechanisms.

Recent requirement smell detection techniques depend on Machine Learning (ML) or Deep Learning (DL) techniques [16, 39, 61]. To boost the performance of such solutions classical NLP or text mining techniques are used in conjugation with the solutions. Machine learning algorithms can classify/detect requirement smells based on the rules and thresholds provided by the experts. There is literature that reports the application of machine learning techniques for the assessment and improvement of requirement quality [61] and for identifying requirement smells [15, 62]. Decision Tree (DT), Feed-Forward Neural Network (FNN), K-Nearest Neighbor (KNN), Logistic Regression (LR), Naïve Bayes (NB), Random Forest (RF), and Support Vector Machine (SVM) are among machine learning algorithms used for requirement smell detection [17, 63-65] Combining DL with conventional NLP provides more robust, generalized and more precise prediction for requirement smell detection [39]. However, utilizing ML/DL techniques for requirement bad smell detection is hampered by insufficient requirement datasets.

3. Research method

The systematic literature review methodology proposed by Kitchenham and Charters [22] was strictly followed to conduct this SLR. The guideline is fine-tuned specifically for performing SLR in software engineering. As per the guidelines, planning, execution, and reporting are the main tasks in the review. This section provides a complete overview of how the planning and execution steps are applied in the research.

3.1. Planning review

In the planning stage, the objectives of the review are reiterated, research questions are defined together with their respective motivations and review protocols are developed and evaluated.

3.1.1. Review objectives and research questions

The main objectives of the study are to identify different kinds of requirement smells and smell detection techniques. Moreover, the study aims to assess the performance of the smell detection tools and the size of the dataset used in selected literature.

Table 4. Research Questions (RQ)

RQ	Research question	Motivation
RQ1	What type of requirement smells are Most commonly detected by the selected studies?	This question aims to know different kinds of requirement smells that are detected by the studies
RQ2	What are the existing requirements for smell identification techniques?	With this question, we aim to assess and understand the techniques for the identification of requirement quality and smell
RQ3	How many requirements are used to assess and evaluate the proposed approaches in the selected studies?	To know whether the researchers have used a small or large number of requirements to evaluate their approach
RQ4	What is the overall performance of requirement smell detection mechanisms?	This question intends to find out the best-performing requirement smell detection techniques
RQ5	What are the tools/libraries/APIs used or produced in the selected study for requirement smell detection?	This question intends to find out the tools, libraries, or Application Programming Interfaces (APIs) used for smell detection

No SLR has been conducted so far on textual requirement smells and their detection methods. To guide the overall direction of the systematic review process Research Questions (RQ) presented in Table 4 are defined. Motivation for each question is also provided.

3.1.2. Develop review protocol

Pre-defining review protocol is essential in a systematic review to reduce the risk and possibility of research biases. The review protocol defines the procedures and rudimentary research processes that are followed during the systematic review. Development of a search strategy (digital libraries and search terms), establishment of inclusion and exclusion criteria, formation of a quality assessment checklist, development of data extraction tools, and identification of study synthesis techniques are part of the definition of review protocol.

Digital libraries. The main databases selected to execute the search string are shown in Table 5. We selected those sources because the databases are prominent academic databases and the quality of software engineering and computer science-related papers published in the databases are proven to have good quality [66].

Table 5. Digital libraries used in search process

Source	Link
ACM Digital Library	**https:/dl.acm.org/**
Springer Link	**https:/link.springer.com/**
ScienceDirect	**https:// ScienceDirect.google.com/**
Google Scholar	**https:/scholar.google.com/**
IEEE Xplore	**https:/ieeexplore.ieee.org/**

Search query. We have used a search string that includes a set of keywords related to software requirements smell and quality. It helps in the completeness of the search studies. After selecting different string options, “AND” and “OR” operators are applied to interconnect those keywords or strings. Then the following string is selected as a general search string:

(“Software requirement” OR “requirement”) AND (“smell” OR “bad smell” OR “quality” OR “ambiguity”) AND (“detection” OR “identification” OR “analysis”)

The search string is customized for databases that suggest different query formats as depicted in Table 6.

Table 6. Search string for the databases

Database	Search string
IEEE	((software requirement smell) OR requirement smell) OR requirement quality) OR (requirement bad smell) AND (detection OR identification OR analysis))
Springer	“requirement” AND “smell” OR “software” AND “requirement” AND “smell” OR “requirement” AND “bad” AND “smell” OR “requirement” AND “quality” AND “detection” OR “identification” OR “analysis”
ACM	(+requirement + smell+ detection) OR (requirement + bad + smell + detection) OR (requirement + bad + smell+ dentification) OR (software + requirement + quality + analysis)

Study selection. The inclusion criteria and exclusion criteria listed below are defined and applied for the selection of the primary studies. The criteria are vital to reduce bias, to reduce search space, and to select only relevant scientific literature.

Inclusion Criteria (IC):

- The studies were published from 2010 to 2023.
- The study was written in English and is open access.
- The studies which are journals and conference papers.
- The studies focused on software requirement smell and requirement quality.

Exclusion Criteria (EC):

- The study is a secondary and tertiary study.
- The study before 2010 was not included in this review.
- The study is an opinion article, workshop, or magazine.

The study selection has been made in four phases. In each phase, inclusion and exclusion criteria were used to minimize the number of studies to select relevant studies (Table 7).

Table 7. Phases for study selection

Phase	Inclusion	Exclusion
1	IC1: The studies were published from 2010 to 2023	EC1: The study was published before 2010
2	IC2: The studies were published in English and are open-access	EC2
3	IC3: The study is a journal or conference paper	EC3: The study is secondary or tertiary
4	IC4: The focused-on-study software requirement smells and requirement quality	EC4: The study is an opinion article, workshop, or magazine

Study Quality Assessment (QA). The quality assessment of the papers was made using a checklist to rate and evaluate the quality of each study identified in the previous step.

- QA1: Were the objectives of the study clearly stated?
- QA2: Did the study describe the main research problem?
- QA3: Were the methodologies of the study clearly stated?
- QA4: Were the study findings evaluated and results discussed?
- QA5: Are limitations, negative findings, or threats to validity explained?
- QA6. Did the study point out any future research direction?
- QA7: Were the detected requirements smells classified based on a criterion?

To evaluate the quality of selected studies, we have used six quality assessment criteria. The quality assessment criteria are needed to define quality criteria and to minimize bias while ensuring the external and internal validity of selected studies. Quality checklists are used as proposed by [22] and other checklists are added according to the research questions.

Data extraction form. The list in Table 8 contains a data extraction form that describes the data that have been extracted from selected papers. These data are summarized and tabulated in a way that can serve the objectives of this SLR. The extracted data are: study identifier, article title, the article source, publication venue, publication year, author/s, and objectives.

Table 8. Data extraction form

Element	Description	Related RQ
Title	Title of the selected study	RQ2
Authors	Authors of the selected study	-
Years	The year the selected study was published	-
Source	The collection in which the study was published	-
Publication type	Type of publication, e.g., journal or conference paper	-
Keywords	Keywords of the study	All
Abstract	Abstract at the beginning of the study	All
Number of requirements	Number of requirements used in the evaluation of the selected study	RQ3
Source requirements	The source the requirements are obtained	RQ4
Requirement smell type	The type of requirement smell detected	RQ1
NLP technique	Different kinds of NLP techniques are employed to identify smells or quality indicators from software requirements if the study employs NLP	RQ1
Requirements smell identification techniques/ tool/method.	Requirement smell detection techniques/tool/method reported in the study	RQ5
Performance measure	The metrics used for measuring the performance	RQ4
Performance values	The achieved performance measurement value	RQ4

Data synthesis techniques. To summarize and present the knowledge gained from the existing studies, tabulation, textual descriptions, and visual diagrams are used.

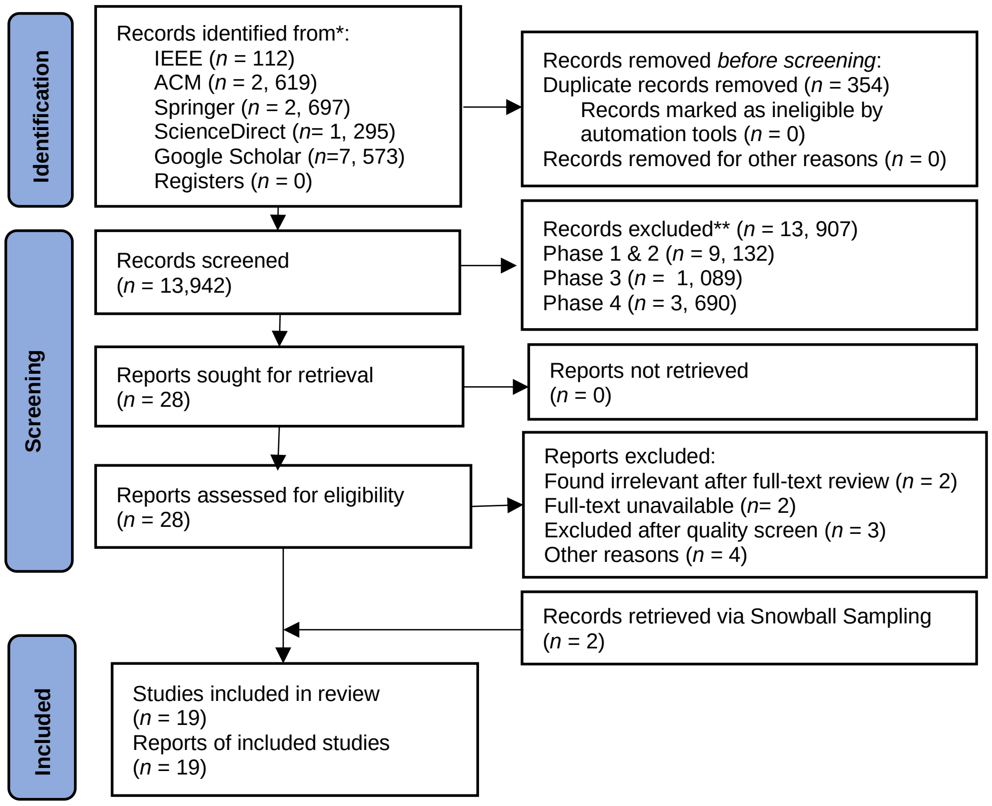
3.2. Conducting the review

This section discusses the execution of the systematic review. The developed review protocol is executed based on the plan. That is, a search is conducted, primary studies are selected, a data extraction form is filled, the quality of the studies is assessed and data synthesis is performed.

3.2.1. Study search and selection

The selection of the primary studies on requirement smell detection is limited to papers published in the span of thirteen years from 2010 up to 2023 by the scope of this study. The search is made on the five digital libraries: ACM digital library, Springer, IEEE Xplore, Google Scholar, and Science Direct. The appropriate search query is applied to each database and the search result is recorded in a spreadsheet and the reference management tool Mendeley.

Repetitive articles and publications that are not related to the subject area are removed from screening. Next, the papers were distributed to two reviewers, and the screening was performed based on the title and abstract of the papers. At this stage, the application of inclusion and exclusion criteria has resulted in candidate papers for analysis. The papers are downloaded and the full text is reviewed by each reviewer independently. Journal and conference articles that are found irrelevant by both reviewers are discarded. The reviewers resolved most of the conflicts through consensus. In a few instances, the third reviewer is involved in resolving the disagreement. For some papers full text is not available and hence such papers are rejected from the study. Papers that are just reviews and do not make any kind of evaluation of the proposed solutions are not included in the study. Papers that deal with smell detection on different form requirements (for instance, goal modeling [14], and use case descriptions [13]) are also removed. There are also papers removed after quality assessment. Two papers are added to the selected paper list after following backward and forward snowball sampling guidelines for SLR in software engineering [67] on the eligible seventeen papers. Nineteen studies were finally selected from the five electronic databases. Fig. 1 displays the final result of the selection process.



Identification
Screening
Included
Records identified from*: IEEE (n = 112) ACM (n = 2, 619) Springer (n = 2, 697) ScienceDirect (n= 1, 295) Google Scholar (n=7, 573) Registers (n = 0)
Records removed before screening: Duplicate records removed (n = 354) Records marked as ineligible by automation tools (n = 0) Records removed for other reasons (n = 0)
Records screened (n = 13,942)
Records excluded** (n = 13, 907) Phase 1 & 2 (n = 9, 132) Phase 3 (n = 1, 089) Phase 4 (n = 3, 690)
Reports sought for retrieval (n = 28)
Reports not retrieved (n = 0)
Reports assessed for eligibility (n = 28)
Reports excluded: Found irrelevant after full-text review (n = 2) Full-text unavailable (n= 2) Excluded after quality screen (n = 3) Other reasons (n = 4)
Records retrieved via Snowball Sampling (n = 2)
Studies included in review (n = 19) Reports of included studies (n = 19)

Fig. 1. PRISMA flowchart showing selection and inclusion of the studies in the review

3.2.2. Study quality assessment

The quality assessment of the papers is executed based on the predetermined quality assessment criteria. Quality assessment rate 1 indicates that the corresponding quality assessment question was answered, whereas 0 indicates the corresponding quality assessment was not answered, and 0.5 indicates the corresponding quality assessment question was answered partially. Then the threshold is decided to be 5 to show that the candidate study has fulfilled more than half of the quality assessment criteria. Therefore, studies with a total quality assessment score of less than 5 were excluded from this SLR. The quality scores of selected papers are shown in Table 9. Each of the quality assessment questions has received more than half responses that favor the assessment criteria. QA4 has received the highest score (92.11%) among the criteria.

That is almost all selected studies have made some kind of evaluations and discussions on the study results. Some studies fail to classify the detected requirement smells resulting 65% quality score for QA7. Quality score of 71.05% is registered for QA3 as some studies fail to describe the methodology followed clearly. The quality score of other quality assessment criteria can be viewed in Fig 2.

3.2.3. Data extraction and synthesis

The data extraction process was executed considering the parameters presented in Table 8 (Data extraction form) to find information important to answer the research questions. Thematic analysis [68], an approach based on relationships and recurring patterns, was used to synthesize the identified themes related to requirements validation. Thematic analysis, which relies on relationships and recurring patterns, was employed to mingle the discerned themes concerning requirements smells using the procedures stated in [68].

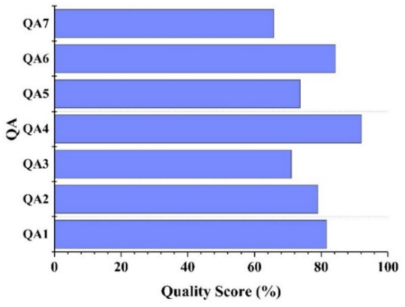


Fig. 2. Percentage scores for the quality assessments of the studies

Table 9. Studies and their quality scores

Study ID	QA1	QA2	QA3	QA4	QA5	QA6	QA7	Total
P1	1	1	0.5	1	0	1	1	5.5
P2	1	1	1	1	1	1	0	6
P3	0.5	1	1	1	1	0	1	5.5
P4	1	1	0.5	1	1	0.5	0	5
P5	1	0.5	1	1	1	1	1	6.5
P6	1	0.5	0.5	1	0.5	1	0.5	5
P7	1	0.5	1	1	1	1	1	6.5
P8	1	1	1	1	1	1	0	6
P9	0.5	1	0.5	0.5	1	0.5	1	5
P10	0.5	1	1	1	0.5	1	0	5
P11	1	0.5	0.5	0.5	1	1	1	5.5
P12	0.5	1	0.5	1	0	1	1	5
P13	1	1	0.5	0.5	0	1	1	5
P14	0.5	1	1	1	1	0	1	5.5
P15	0.5	0.5	1	1	1	1	0	5
P16	1	1	0.5	1	1	1	1	6.5
P17	1	0.5	0.5	1	0	1	1	5
P18	0.5	0.5	0.5	1	1	1	1	5.5
P19	1	0.5	0.5	1	1	1	0	5

Each select paper was carefully scrutinized to understand the topics very well and then the information that answers the research questions was extracted and coded. Subsequently, the codes were translated to hierarchical themes, the relationship among themes was investigated and finally, the trustworthiness of the interpretations was assessed and approved. Two reviewers worked independently in the extraction of the data independently using the thematic analysis procedures. While merging the results whenever there was disagreement an invited reviewer was involved. Table 10 reports the final selected primary studies that were used in this review with their publication venue/type, year, and source. The organizational structure of most selected studies was almost similar, they first dealt with the purpose of software requirement then they dealt with the problem with quality of requirement or smells finally most of them tried to suggest their researched solution.

Table 10. Selected Studies within their Information

Paper ID	Title	Year	Paper venue	Publication type	Reference
P1	NERO: a text-based tool for content annotation and detection of smells in feature requests	2020	IEEE	Conference	[38]
P2	A methodology for the classification of quality of requirements using machine learning techniques	2015	Science Direct	Journal	[16]
P3	Application of machine learning techniques to the flexible assessment and improvement of requirements quality	2020	Springer	Journal	[61]
P4	Using NLP to detect requirements defects: an industrial experience in the railway domain	2017	Springer	Journal	[53]
P5	Rapid requirements checks with requirements smells: two case studies henning	2014	ACM	Journal	[4]
P6	Detecting requirements smells with deep learning: experiences, challenges and future work	2021	IEEE	Journal	[39]
P7	Rapid quality assurance with requirements smells	2017	Science Direct	Journal	[37]
P8	Quality assessment method for software requirements specifications based on document characteristics and its structure	2015	IEEE	Conference	[56]
P9	Ambiguity detection: towards a tool explaining ambiguity sources	2010	Springer	Journal	[57]
P10	On the ability of lightweight checks to detect ambiguity in requirements documentation	2017	Springer	Journal	[58]
P11	Automated smell detection and recommendation in natural language requirements	2023	Google Scholar	Journal	[8]
P12	Classification and prioritization of requirements smells using machine learning techniques	2023	IEEE	Conference	[17]
P13	A Method of ambiguity detection in requirement specifications by using a knowledge dictionary	2022	Science Direct	Conference	[60]
P14	Ambiguous software requirement specification detection: an automated approach	2018	IEEE	Conference	[53]
P15	Using domain-specific corpora for improved handling of ambiguity in requirements	2021	IEEE	Conference	[54]
P16	The design of sree – a prototype potential ambiguity finder for requirements specifications and lessons learned	2013	Springer	Conference	[59]
P17	Score-based automatic detection and resolution of syntactic ambiguity in natural language requirements	2020	IEEE	Conference	[55]
P18	Analysing anaphoric ambiguity in natural language requirements	2011	Springer	Journal	[64]
P19	Automated handling of anaphoric ambiguity in requirements: a multi-solution study	2022	ACM	Conference	[65]

The distribution of the selected papers over the digital libraries is shown in Fig 3. The majority of the papers (69%) are from Springer and IEEE.

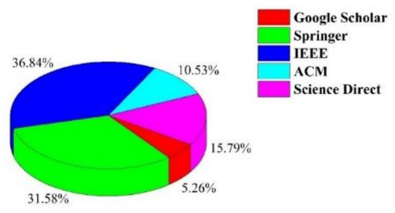


Fig. 3. Selected studies within their source

The distribution of the studies extracted from the five databases over the years (2010-2023) is shown in Fig 4. The increase in the number of studies over time depicts the attention given to requirement smells.

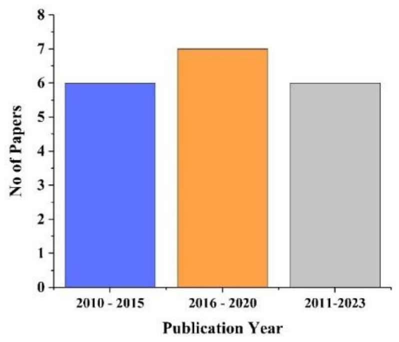


Fig. 4. Distribution of selected studies over the years

The word cloud formed by taking the titles of selected studies using **https://tagcrowd.com** is shown in Fig 5. Non-common English words repeated at least twice appear in the diagram. The word cloud shows the concepts focused on by this study and it helped us to shape the title and keywords of the study.

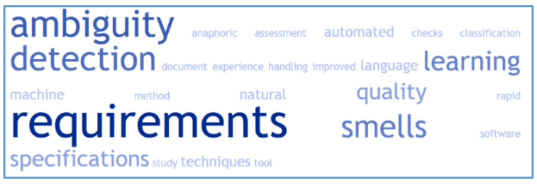


Fig. 5. Word cloud from the titles primary studies

4. Result and discussions

This section presents the results of the systematic literature review, in which nineteen (19) studies have been discussed to respond to defined research questions.

4.1. Requirement smell types

RQ1: What type of requirement smells are most commonly detected by the selected studies?

Different kinds of requirement smells are considered by the selected studies and needless to say each smell has its own undesirable impact on the activities of the requirements and software development lifecycle. Most of the selected studies introduced an automated detection of requirement smells or quality problems. After fetching requirement smells detected from the selected papers we have grouped them into five categories: morphological, lexical, analytical, relational, and incompleteness & language requirement smells. The detection mechanism may target specific requirement smell or multiple smells. For instance, P15 is dedicated to the detection of speculative sentences while P2 is involved in the detection of many types of smells. As shown in Table 11 most of the studies are able to detect requirement smells from more than one requirement smell category. Especially, P1, P2, P3, P4, P8 and P11 are able to detect smells from three or more requirement smell groups. Many studies are dedicated to only one type of the requirement smells: P6, P10, P13, P14, P15, P17, P18, & P19.

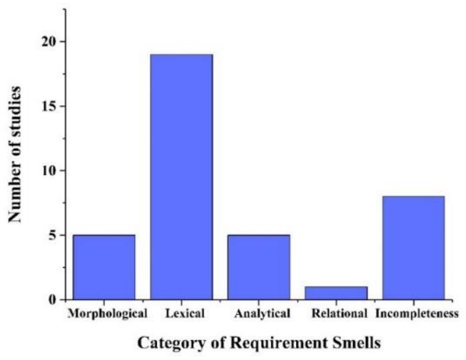


Fig. 6. Category of detected requirement smells by the selected studies

Fig. 6 shows the number of studies against the categories of the requirement smells. Lexical requirement smells are addressed by all selected studies. The main reason could be lexical requirement smells are a direct manifestation of natural language problems. On the other hand, relational requirement smells are the least studied group of smells. To detect relational requirement quality problems, more than a word, phrase or statement has to be scrutinized as relational requirement smells consider more than one statement.

Considering individual smells, we have identified the most frequently detected requirement smells. For this purpose, we have grouped smells that have the same meaning but are described in different wordings. For instance, the use of coordinators in different studies is described as coordinating conjunction, connective terms, non-atomic requirements, or the use of multiple connectors. All of them describe requirement amalgamation (describing more than one requirement in a single statement) [9]. Statements should include a single requirement with no use of conjunctions that describe multiple requirements. Fig. 7 shows the top six frequently detected requirement smells by the selected studies. Ambiguity is detected more than twenty times. There are studies that have detected more than one type of ambiguity: P1, P4, P9, P15 & P17. The use of coordinators, vague pronouns, non-verifiable terms, passive voices, and subjective languages are the next smells in the rank.

Table 11. Detected requirement smells collected from selected studies

Paper	Requirement smell type	Morpho-logical	Lexical	Analytical	Relational	Incompleteness and language
P1	Vagueness, weakness, generality, coordination ambiguity, referential ambiguity, passive voice, missing description, missing condition, unreadability, partial content	x	x	x		x
P2	Ambiguous expression, verbal tense and mood, connective terms, high dependencies of a requirement, design terms, domain terms, incomplete listing/incompleteness expressions, degree of nesting, punctuation, too many wording/paragraphs, imprecise terms, speculative expression, unreadability	x	x		x	x
P3	Too many paragraphs, an excessive number of words, unreadability, incorrect punctuation, use of multiple connectors, use of negative expressions, use of pseudocode and control-flow expression, non-explicit requirements (implicit), ambiguous, incomplete enumerations, speculative expressions, justifications in the requirement (rationale), requirement expressing a solution (design), do not have at least one imperative verb (imperative), the non-assertive requirement (conditional), use of passive voice, use of a large number of domain concepts, too many domain verbs	x	x	x		
P4	Anaphoric ambiguity, coordination ambiguity, vague terms, use of modal adverbs, passive voice, excessive length, missing condition, missing unit of measurement, missing reference, undefined terms	x	x			x
P5	Ambiguous adverbs & adjectives, vague pronouns, subjective language smell, comparative phrases, superlatives, negative statements, non-verifiable terms, loopholes smell, incomplete references		x			x
P6	Subjective language, ambiguous adverbs and adjectives, superlatives, comparatives, and vague pronouns		x			
P7	Subjective language, ambiguous adverbs, and adjectives, loopholes, open-ended, non-verifiable terms, superlatives, comparatives, negative statements, vague pronouns, incomplete references		x			x

Table 11 (c o n t i n u e d)

Paper	Requirement smell type	Morpho-logical	Lexical	Analytical	Relational	Incompleteness and language
P8	Ambiguous word, abbreviation, acronym, subjectivity, optional, modal verb, more than one verb, pronoun, demonstrative adjective, quantifier, coordinating conjunction, punctuation	x	x			x
P9	Lexical ambiguity, syntactic ambiguity, semantic ambiguity, pragmatic ambiguity, vagueness, language error, passive voice, ambiguous adjectives and adverbs		x	x		
P10	Ambiguity		x			
P11	Non-atomic requirement, incomplete requirement, not requirement, incomplete condition, incomplete system response, passive voice, incorrect order requirement, coordination ambiguity, not precise verb		x	x		x
P12	Superlative phrase, comparative phrase, subjective language, vague pronoun, loopholes, ambiguous adverb adjective stive, negative statements, open-ended non-verifiable term, passive voice		x	x		
P13	Ambiguity		x			
P14	Ambiguity		x			
P15	Coordination ambiguity and prepositional-phrase attachment ambiguity		x			
P16	Use of continuance, use of coordinator, use of directives, incomplete (missing), use options, use of the pronoun, use of quantifier, use of vague terms and symbols, weakness, use of plural nouns		x			x
P17	Coordination, attachment, & analytical ambiguities		x			
P18	Nocuous anaphoric ambiguity		x			
P19	Anaphoric ambiguity		x			

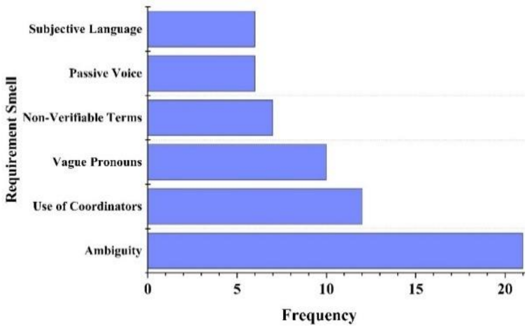


Fig. 7. Frequently detected requirement smells

On the other hand, many requirements are detected by just a single study. Some of such requirements are missing unit of measurement – P4, high dependencies of a requirement – P2, missing description – P1, undefined terms – P4, incorrect requirement order – P11, and use of plural nouns – P16. Moreover, we have found smells like user interface details smells [69], terms that imply totality (for example "all", "always", "never", and "every") [9], test clones, and long tests [70] from other literature but are not addressed by any of the selected studies.

4.2. Requirement smell identification techniques

RQ2: What are the existing requirements for smell identification techniques?

The requirement smell detection techniques reported by the selected studies can be grouped into five as natural language processing, machine learning, natural language processing, and machine learning, rule-based, and knowledge dictionary-based techniques. NLP-based techniques involve processing or linguistic analysis of textual requirements to find smells from the requirements using detailed handwritten rules [71]. Machine learning techniques take a raw set of requirements, possibly pre-annotated by experts, and emulate experts by learning from the input data [61]. If the approach applies explicitly stated NLP and ML tasks, we grouped the detection technique as NLP and ML. If a selected paper applies IF-THEN-like rules to pinpoint a smell that is rule rule-based technique. A knowledge dictionary built from transitive verbs and their objects is also used to detect for detecting ambiguities P13.

Most of the required bad smell detection approaches employed NLP techniques such as POS tagging, morphological analysis, dictionary-based, lemmatization, parsing, etc., see Fig. 8. Selected Studies that employee NLP include P1, P4, P5, P7, P11, P15, and P17. Natural language processing techniques together with machine learning techniques are also applied in some studies P6, P12, P18 & P19. Rule-based techniques are applied to detect poor quality requirements: P8, P9, P10 and P16. Furthermore, machine Learning approaches were introduced by P2, P3, and P14 for the assessment and classification of requirement quality to check whether the requirement has a good or bad quality. These techniques use requirements written in natural languages as an input for the learning algorithms but no explicit NLP techniques are reported in the papers. Deep learning algorithm also applied by P6 for detection of multiple classes of requirement smells.

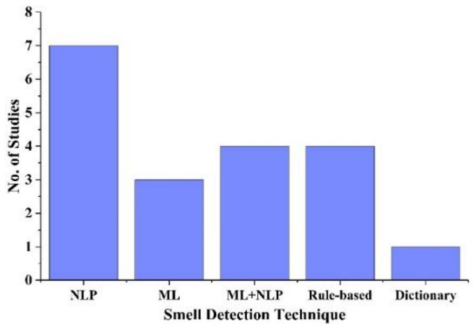


Fig. 8. Requirement smell detection mechanism

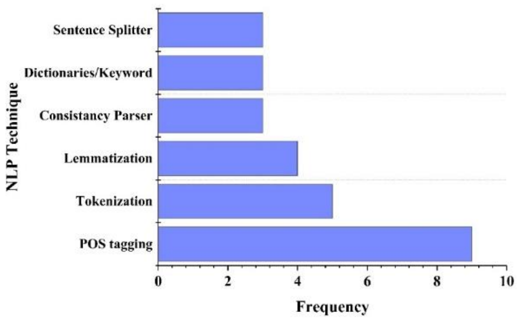


Fig. 9. NLP Techniques and frequency of application

The NLP techniques were used for the identification of quality defects and ambiguity, classification and clustering of large collections of natural language requirements, and so on. Fig. 9 shows the most commonly used NLP techniques in the selected study. As it is possible to observe, POS tagging is used by many selected studies i.e., it is used in nine different instances or selected studies applied it. POS tagging which involves labeling words in a statement with their respective POS tags is the most significant text pre-processing task for NLP activities [72]. Tokenization, lemmatization, consistency parsing, dictionaries, and sentence splitting are the next most repeatedly used NLP techniques in the selected studies.

4.3. Datasets for requirement smells

RQ3: How many requirements are used to assess and evaluate the proposed approaches in the selected studies?

The requirements collected from software projects are used as a dataset in the selected studies. We consider here the total number of requirements that have been used by each of the selected studies. Some of the studies used a small number of requirements (10, 100, 126, 293, and 398) to evaluate the performance of their approaches. Other studies reported the number of datasets used in the range of 1000 to 26, 829. Fig. 10 shows the number of requirements used in the studies. We have used a logarithmic scale to resolve visualization issues because a few points of data are much larger than the others [21]. The advised number of requirements depends on the smell detection technique applied. Generally, a large number of datasets are recommended if machine/deep learning techniques are used.

The datasets used by the selected studies could be open source which are available freely to the public or are closed sources which are mostly collected from industrial projects and are not open to access. Except for one paper which failed to mention the source of the data P17, other papers have used data from open source P1, P2, P3, P7, and P18, closed source P4, P5, P8, P9, P10, P11, P13, P14, P15, P16, P17, and P19 or from both sources P6 and P12. 64% of the primary study used requirements whose access is restricted (Fig. 11). The availability of free open-source datasets on requirement smells is vital to conducting more research in the area

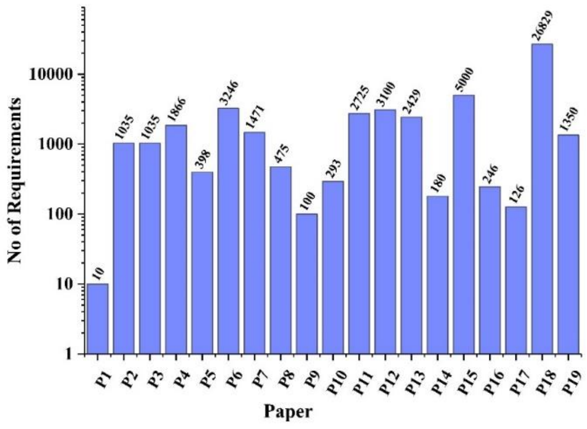


Fig. 10. Number of requirements used in the selected studies

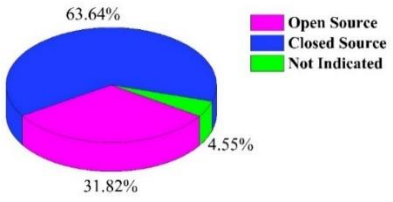


Fig. 11. Sources of datasets

4.4. Performance of required smell detection techniques

RQ4: What is the overall performance of requirement smell detection mechanisms?

A review is made on the performance of requirement smell detection techniques as reported by the selected papers. The performance of the proposed solutions is measured using accuracy, precession, recall, F-measure, and Spearman's ρ. Precession and recall are the most frequently used performance measures, (Fig. 12). Some papers have employed more than one measure. Spearman's ρ measure which measures the strength of the relationship between two variables just like Pearson correlation is used at one instance P1. The performance average value ranges from 50% accuracy P8 to 100% for precession P13 and 100% recall P13 and P19. We don't think comparing the results of the studies is appropriate as each of the studies used a different method and detected different requirement smells.

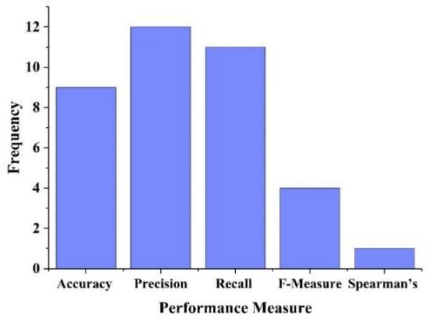


Fig. 12. Performance measures

4.5. Tools for requirement smell detection

RQ5: What are the tools/libraries/APIs used or produced in the selected study?

Authors of the selected papers have employed different types of Computer Aided Software Engineering (CASE) tools, see Table 12. We have categorized the tools into four Requirement Quality Assessment (RQA) tools, ML tools, NLP tools, and other tools. RQA tools are used to assess the quality of requirements and possibly to provide a proposal for resolution. ML tools implement artificial intelligence algorithms that allow a computer to understand and improve without human intervention. NLP tools enable to process of natural languages to pave the way for a computer to interpret, manipulate, and comprehend human languages. Tools that couldn't be categorized in either of the three categories of tools are termed other tools.

Table 12. Tools used for required smell detection

Tool category	Tools
NLP tools	spaCy [73], Stanford CoreNLP [74], AllenNLP [75], DKPro [76], GATE [77], Genia Tagger [78], jWeb1T library [79], JWPL library [80], NLTK package [81, 82], Oxford Advanced Learner's Dictionary [83], QuARS tool [84, 85], RapidMiner [86], Sentence Boundary, Detector [87], Stanza [88], Stanford CoreNLP [89, 90], WordNet tool [91], WS4J library [92], Stanford, Parser [74]
RQA tools	SREE [59], Paska [8], Rimay [93], knowledge dictionary-based tool [49], ambiguity detector2 [57], ambiguity detector1[63], ConQAT [87, 94], NASA ARM tool [95], NERO [38], RQA for extract quality metrics that measure desirable properties of requirements [45], SMELL tool [4], Smella [37], Tactile check [58]
ML tools	Scikit-learn [96], WEKA [97]
Other tools	Stanford Tregex API for Java [98], Wikipedia

As shown in Fig. 13 most studies employ NLP tools (53%) and RQA tools (37%). A limited number of machine learning and other tools are used. Even though requirement engineering is the least tool-supported software development phase, a number of tools are utilized for requirement smell detection.

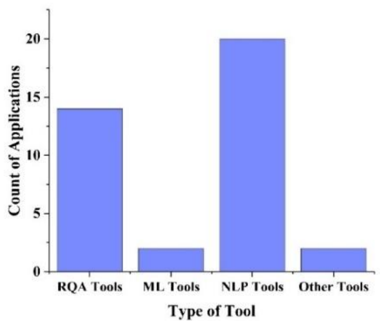


Fig. 13. Types of tools used for requirement smell detection

5. Threats to the validity

The cogency or accuracy of research including systematic literature review is negatively influenced by validity threats. The four common categories of threats to the validity of a research outcome are conclusion, internal, construct, and external validity [99]. In this research, to evaluate and allay the four types of threats to validity, guidelines specifically tuned up for SLR in software engineering are used [100]. We illustrate the threats in terms of the four categories of validity.

External validity on SLR includes restricted time span, incomplete research information in the primary study, paper/database inaccessibility, and primary study generalizability [99]. The SLR investigates papers published in thirteen years of time span and the starting time period goes back to three years from the requirement smell introduced in the doctoral symposium [10]. Each of the selected studies contains information that addresses more than 60% of the research questions and this addresses the primary study's generalizability and incomplete research information in the primary study. The application of the snowball sampling method and using customized search strings reduced inadequate primary studies. The papers are obtained from well-known literature repositories which are frequently used in SLR research in software engineering [2, 18-20].

Internal validity is about a cause-and-effect relationship established in the study supported by evidence [100]. As an SLR research, this research is not concerned about the statistical causal relationship of variables in requirement smells. However, SLR internal validity threats may arise from improper research methods and analysis and coding process researcher biases. To overcome the challenges, the SLR research method which is specifically customized for software engineering [22, 68, 101] is followed and data is collected from well-known databases. To include as many primary studies as possible we have applied forward and backward snowball sampling. The research bias is reduced as two experts are involved in the analysis and conflicts are resolved by inviting a third reviewer.

Construct validity deals with whether the research made the right/intended measures for the concepts under study [100, 102]. In this study, construct validity is maximized by recording all results after a full paper review in Excel by following the review protocol. Inappropriate research questions are one of the threats to construct validity [99]. However, the research questions are evaluated for the characteristics they should possess [103] before structuring the research and reporting.

If a study has conclusion validity, data extraction and interpretation are made properly and hence the data collection process can be repeated resulting in the same result [99]. The conclusion validity is enhanced as we have used systematic review guidelines for software engineering [22, 68] to review the primary studies. Albeit complete elimination of bias is not feasible, the involvement of at least two reviewers for each paper and the invitation of the third reviewer to resolve any disagreement reduces biasedness. The use of Microsoft Excel and Mendeley tackled the threat of duplication. Misclassification of primary study is another threat to conclusion validity which is addressed by following the guidelines stated at [68].

6. Future research directions

Developing software that is highly qualified and competent requires good quality requirements. Writing requirements using textual descriptions makes specifying good quality requirements challenging. Identifying ill-defined requirements written in natural language is difficult as human languages increase the complexity and effort required for software development [10, 39, 89, 104]. Hence, many automated tools and techniques are proposed in the literature. Findings indicate that existing approaches are not enough to identify multiple types of requirements smells. So, taking into account the above consideration, we want to highlight some issues related to requirement smell detection that could be exploited in the future.

- **Utilize machine learning for smell detection** – Studies on requirement smell detection should broadly use machine learning and deep learning techniques since the techniques are found to be effective in solving many types of problems. In fact, 36.84% of the selected studies have used machine learning techniques. However, since machine learning and natural language processing are good companions to solve requirement engineering problems [71] and as the requirements are usually written in natural languages [10], extensive use of the techniques will enable finding out requirement quality indicators that are not easy to detect by using traditional methods.
- **Dataset for requirement smell** – One of the reasons for underutilization and poor performance of machine learning techniques is the lack of sufficient and quality datasets in the area of requirement engineering in general [105]. Defining annotation rules, unbalanced distribution of samples, and very high effort for data processing and data quality improvements are some of the challenges that hinder the application of ML. There are attempts to produce datasets in requirement engineering [106]. However, there is no known dataset for requirement smells. Producing datasets on requirement smells will facilitate different research in the area.
- **Requirement smell detection and type of requirements** – Most studies focus on requirement bad smell detection. Requirements can be written at various levels of detail and as such requirements can be categorized as business, user, and system requirements [107]. The way each type of requirement is written is different.

Consequently, a different way of smell detection approaches should be proposed based on the type of requirements.

- **Types of requirement smells** – Types of requirement smells or quality indicators need further investigation as it may lead to new smell discovery. A catalog of required smells needs to be prepared and the smells need to be defined and classified based on their relatedness.
- **Impact of requirement smells on software project** – The influence of each *requirement* on the later phases of SDL and software projects, in general, needs to be investigated using exhaustive case studies. That is an empirical study on the impact/evolution of smells on the later stage of the software development life cycle enables us to know the exact defilement the smells cause on software projects and the software product itself in the same as it is studied for design smells in [108].
- **Prioritizing requirement smells** – Bad smells are not expected to have a similar impact on software projects. Hence, there shall be a way to prioritize requirement smells based on different criteria and this will help to take appropriate action and to improve the quality of software
- **Beyond detection of the requirement smells** – The investigated studies focus on requirement smell detection mechanisms keeping in mind that once the smells are detected, they can be manually corrected. However, for large and complex software systems, automatic smell resolution solutions need to be proposed. Research should be conducted to find efficient ways to resolve requirement smells.

7. Conclusion

Software development starts with the discovery of requirements. Poor quality requirements or requirements with smells cause the software project to fail. Hence, ensuring requirement quality at the earliest possible stage is an important activity in the process of software development. A requirement smell detected at the maintenance phase may cost 200% more than detecting it at the requirement specification phase. This literature review summarizes existing studies obtained from five known databases and published from 2010 to 2023. The selected studies focused on proposing NLP, machine learning, rule, and knowledge-based requirement smell detection solutions. Precession, accuracy, and recall are evaluation metrics used to evaluate most of the bad smell detection tools and the performance measures vary in the range of 50% to 100%. The detection mechanisms employed CASE tools like requirement quality analyzer tool, machine learning tool, and NLP tools. Various types of requirement smells are detected in the selected primary studies. Requirement problems that are related to the ambiguity of words, phrases, and statements are given the highest attention. Finally, the role of machine/deep learning for requirement smell detection as well as proposing requirement-specific smell detection types needs to be investigated. Furthermore, tasks such as classifying and prioritizing requirement smell and providing a requirement smells dataset are assignments for software engineering researchers in industry and academia.

References

1. J i t n a h, D., J. H a n, P. S t e e l e. Software Requirements Engineering : An Overview 1 Introduction 2 Preliminaries. – Penins Sch. Comput. Inf. Technol. Monash. Univ., 1995, pp. 1-20.
2. A h m a d, A., C. F e n g, M. K h a n, A. K h a n, A. U l l a h, S. N a z i r, A. T a h i r. A Systematic Literature Review on Using Machine Learning Algorithms for Software Requirements Identification on Stack Overflow. – Security and Communication Networks, Vol. **2020**, 2020. DOI: 10.1155/2020/8830683.
3. 830-1993, Recommended Practice for Software Requirements Specification. IEEE Computer Society. – Software Engineering Standard Committee of the IEEE Std Computer Society. **Revision**, 1998. p. 32.
4. F e m m e r, H., D. M. F e r n á n d e z, E. J u e r g e n s, M. K l o s e, I. Z i m m e r, J. Z i m m e r. Rapid Requirements Checks with Requirements Smells: Two Case Studies. – In: Proc. of 1st International Workshop on Rapid Continuous Software Engineering (RCoSE 2014), 2014, pp. 10-19. DOI: 10.1145/2593812.2593817.
5. S i t e s, M., R. W. S e l b y. Software Engineering: Barry W. Boehm’s Lifetime Contributions to Software Development, Management, and Research. – Wiley-IEEE Press, 2007, pp. 1-13. DOI: 10.1109/9780470187562.ch8.
6. N a e e m, A., Z. A s l a m, M. A. S h a h. Analyzing Quality of Software Requirements ; A Comparison Study on NLP Tools. – In: Proc. of 25th International Conference on Automation and Computing (ICAC’19), 2019, No September, pp. 1-6. DOI: 10.23919/IConAC.2019.8895182.
7. K o c e r k a, Ī. J., Ī. M i c h a, Ī. A. G a. Analysing Quality of Textual Requirements Using Natural Language Processing. – A Literature Review, 2018, pp. 876-880. DOI: 10.1109/MMAR.2018.8486143.
8. V e i z a g a, A., S. Y. S h i n, L. C. B r i a n d. Automated Smell Detection and Recommendation in Natural Language Requirements. – IEEE Transactions on Software Engineering, 2024, pp. 1-26. DOI: 10.1109/TSE.2024.3361033.
9. Xplore I. International Standard ISO/IEC/IEEE Systems and Software Engineering. – Engineering, Vol. **2018**, 2018.
10. F e m m e r, H. Reviewing Natural Language Requirements with Requirements Smells – A Research Proposal – Categories and Subject Descriptors. – In: Proc. of 11th International Doctoral Symposium on Empirical Software Engineering (IDoESE’13 at ESEM’13), 2013.
11. A n g a r a, J., S. P r a s a d, G. S r i d e v i. DevOPs Project Management Tools for Sprint Planning, Estimation and Execution Maturity. – Cybernetics and Information Technologies, Vol. **20**, 2020, No 2, pp. 79-92.
12. M e a d, N. R., T. S t e h n e y. Security Quality Requirements Engineering (SQUARE) Methodology. –- In: Proc. of 2005 Workshop on Software Engineering for Secure Systems – Building Trustworthy Applications (SESS’2005), 2005, pp. 1-7. DOI: 10.1145/1083200.1083214.
13. S e k i, Y., S. H a y a s h i, M. S a e k i. Detecting Bad Smells in Use Case Descriptions. – In: Proc. of 27th IEEE International Requirements Engineering Conference (RE’19), 2019, pp. 98-108. DOI: 10.1109/RE.2019.00021.
14. A l t u r a y e i f, N., I. A b d u r a h m a n, B. F a i s a l. Detection of Linguistic Bad Smells in GRL Models : An NLP Approach. – In: Proc. of ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C’23), 2023, pp. 318-327. DOI: 10.1109/MODELS-C59198.2023.00062.
15. C h a i t h r a, P., S. N a y a k. Machine Learning Technique for Identifying Ambiguities of in Software Requirements. – Turkish Journal of Computer and Mathematics Education, Vol. **12**, 2021, No 11, pp. 6852-6857. DOI: 10.17762/turcomat.v12i11.7159.
16. P a r r a, E., C. D i m o u, J. L l o r e n s, V. M o r e n o, A. F r a g a. A Methodology for the Classification of Quality of Requirements Using Machine Learning Techniques. – Information and Software Technology, Vol. **67**, 2015, pp. 180-195. DOI: 10.1016/j.infsof.2015.07.006.

17. B e r h a n u, F., E. A l e m n e h. Classification and Prioritization of Requirements Smells Using Machine Learning Techniques. – In: Proc. of International Conference on Information and Communication Technology for Development for Africa (ICT4DA'23), 2023, pp. 49-54. DOI: 10.1109/ICT4DA59526.2023.10302263.
18. A l, A., S. H a m o u d, A. M o h a m m a d. Bad Smell Detection Using Machine Learning Techniques : A Systematic Literature Review. – Arabian Journal for Science and Engineering, Vol. **45**, 2020, No 0123456789, pp. 2341-2369. DOI: 10.1007/s13369-019-04311-w.
19. S a b i r, F., F. P a l m a, G. R a s o o l, N. M o h a. A Systematic Literature Review on the Detection of Smells and Their Evolution in Object-Oriented and Service-Oriented Systems. – Journal of Software: Practice and Experience, 2019, No July 2018, pp. 3-39. DOI: 10.1002/spe.2639.
20. A l j e d a a n i, W., A. P e r u m a, A. A l j o h a n i, M. A l o t a i b i, M. W. M k a o u e r, A. O u n i, C. D. N e w m a n, A. G h a l l a b, S. L u d i. Test Smell Detection Tools: A Systematic Mapping Study. – In: Proc. of 25th International Conference on Evaluation and Assessment in Software Engineering, 2021, pp. 170-180. DOI: 10.1145/3463274.3463335.
21. M a y r a-A l e j a n d r a, C a s t i l l o-M o t t a, R u b é n-D a r í o, D o r a d o-C ó r d o b a, C é s a r-J e s ú s. Pardo-Calvache. Systematic Mapping of the Literature on Smells in Software Development Requirements. – Revista Facultad de Ingeniería, Vol. **32**, 2023, No 63 pp. 0-3. DOI: 10.19053/01211129.v32.n63.2023.15233.
22. K i t c h e n h a m, B., S. M. C h a r t e r s. Guidelines for Performing Systematic Literature Reviews. – In: Software Engineering, 2007, No October 2021.
23. K i t c h e n h a m, B., L. M a d e y s k i, S. M e m b e r, D. B u d g e n. SEGRESS: Software Engineering Guidelines for Reporting Secondary Studies. – IEEE Transactions on Software Engineering, Vol. **49**, 2023, No 3, pp. 1273-1298. DOI: 10.1109/TSE.2022.3174092.
24. S h a r m a, T., D. S p i n e l l i s. Definitions of a Software Smell. **https://zenodo.org/records/1066135**
25. C h a t z i g e o r g i o u, A., A. M a n a k o s. Investigating the Evolution of Code Smells in Object-Oriented Systems. – Innovations in Systems and Software Engineering, Vol. **10**, 2014, pp. 3-18. DOI: 10.1007/s11334-013-0205-z.
26. S h a r m a, T., M. F r a g k o u l i s, D. S p i n e l l i s. Does Your Configuration Code Smell ? – In: Proc. of 13th International Conference on Mining Software Repositories, 2016, pp. 189-200. DOI: 10.1145/2901739.2901761.
27. B e c k, K., J. B r a n t, W. O p d y k e. Refactoring: Improving the Design of Existing Code. – In: Addison-Wesley Professional, 2018.
28. G a r c i a, J., D. P o p e s c u, G. E d w a r d s, N. M e d v i d o v i c. Identifying Architectural Bad Smells. – In: Proc. of 13th European Conference on Software Maintenance and Reengineering, 2009, pp. 255-258. DOI: 10.1109/CSMR.2009.59.
29. B o u h o u r s, C., H. L e b l a n c, C. P e r c e b o i s. Bad Smells in Design and Design Patterns. – The Journal of Object Technology, Vol. **8**, 2010, No 3, pp. 43-63. DOI: 10.5381/jot.2009.8.3.c5.
30. R a j k o v i c, K., E. E n o i u. NALABS: Detecting Bad Smells in Natural Language Requirements and Test Specifications. – ArXiv Preprint ArXiv:220205641, 2022, pp. 8-10. DOI: 10.48550/arXiv.2202.05641.
31. S h a r m a, T., D. S p i n e l l i s. A Survey on Software Smells. – The Journal of Systems & Software, Vol. **138**, 2018, pp. 158-173. DOI: 10.1016/j.jss.2017.12.034.
32. A b u h a s s a n, A. Software Smell Detection Techniques. – A Systematic Literature Review, 2021, No September 2019, pp. 1-48. DOI: 10.1002/smr.2320.
33. A l m e i d a, D., J. C. C a m p o s, J. S a r a i v a, J. C. S i l v a. Towards a Catalog of Usability Smells. – In: Proc. of 30th Annual ACM Symposium on Applied Computing, 2014, pp. 175-181. DOI: 10.1145/2695664.2695670.
34. P a l m a, F., N. M o h a. A Study on the Taxonomy of Service Antipatterns. – In: Proc. of 2nd IEEE International Workshop on Patterns Promotion and Anti-Patterns Prevention (PPAP'15), 2015, pp. 5-8.
35. P i v e t a, E. K., M. H e c h t, A. M o r e i r a, M. S. P i m e n t a, J. A r a ú j o, P. G u e r r e i r o, R. T. P r i c e. Avoiding Bad Smells in Aspect-Oriented Software. – In: Proc. of 19th International Conference on Software Engineering and Knowledge Engineering (SEKE'07), 2007, pp. 81-86.

36. V a l e, G., E. F i g u e i r e d o, R. A b i l i o, H. C o s t a. Bad Smells in Software Product Lines: A Systematic Review. – In: Proc. of 8th Brazilian Symposium on Software Components, Architectures and Reuse (SBCARS'14), 2014, pp. 84-94. DOI: 10.1109/SBCARS.2014.21.
37. F e m m e r, H., D. M é n d e z F e r n á n d e z, S. W a g n e r, S. E d e r. Rapid Quality Assurance with Requirements Smells. – Journal of Systems and Software, Vol. **123**, 2017, pp. 190-213. DOI: 10.1016/j.jss.2016.02.047.
38. M u, F., L. S h i, W. Z h o u, Y. Z h a n g, H. Z h a o. NERO: A Text-Based Tool for Content Annotation and Detection of Smells in Feature Requests. – Proceedings of IEEE International Conference on Requirements Engineering, Vol. **2020**-**August**, 2020, pp. 400-403. DOI: 10.1109/RE48521.2020.00056.
39. H a b i b, M. K., S. W a g n e r, D. G r a z i o t i n. Detecting Requirements Smells with Deep Learning: Experiences, Challenges and Future Work. – Proceedings of IEEE International Conference on Requirements Engineering, Vol. **2021**-**September**, 2021, pp. 153-156. DOI: 10.1109/REW53955.2021.00027.
40. G e n t i l i, E m a n u e l e, D. F a l e s s i. Characterizing Requirements Smells. – In: Proc. of International Conference on Product-Focused Software Process Improvement. Submitted on 17 April 2024. DOI: 10.48550/arXiv.2404.11106.
41. K a t a s o n o v, A., M. S a k k i n e n. Requirements Quality Control : A Unifying Framework. – Requirements Engineering, 2006, pp. 42-57. DOI: 10.1007/s00766-005-0018-1.
42. M i c h, L., M. F r a n c h, P. L. N o v i I n v e r a r d i. Market Research for Requirements Analysis Using Linguistic Tools. – Requirements Engineering, Vol. **9**, 2004, No 2, pp. 151-151. DOI: 10.1007/s00766-004-0195-3.
43. S e k i, Y., S. H a y a s h i, M. S a e k i. Cataloging Bad Smells in Use Case Descriptions and Automating. – In: Proc. of 2019 IEEE 27th IEEE International Requirements Engineering Conference, 2022, No 5, pp. 849-863. DOI: 10.1587/transinf.2021KBP0008.
44. S i n g h, G., J. C. C a r v e r. A Systematic Literature Review to Identify and Classify Software Requirement Errors. – Information and Software Technology, Vol. **51**, 2009, No 7, pp. 1087-1109. DOI: 10.1016/j.infsof.2009.01.004.
45. G e´n o v a, G., J. M. F u e n t e s, J. L l o r e n s, O. H u r t a d o, V. M o r e n o. A Framework to Measure and Improve the Quality of Textual Requirements. – Requirement Engineering, Vol. **16**, 2013, pp. 25-41. DOI: 10.1007/s00766-011-0134-z.
46. K r o g s t i e, J., O. I. L i n d l a n d. Towards a Deeper Understanding of Quality in Requirements Engineering Domain Quality Appropriateness Appropriatenes. – Model I Quality Language I Pragmatic I Interpretation Audience Appropriateness, Vol. **932**, 1995, pp. 82-95. DOI: 10.1007/978-3-642-36926-1_7.
47. S a a v e d r a, R., L. B a l l e j o s, M. A l e. Software Requirements Quality Evaluation: State of the Art and Research Challenges. – In: Proc. of 14th Argentine Symposium on Software Engineering, 2013, pp. 240-257.
48. Ö z k a n, D., A. M i s h r a. Agile Project Management Tools: A Brief Comprative View. – Cybernetics and Information Technologies, Vol. **19**, 2019, No 4, pp. 17-25.
49. Z a k e r i-N a s r a b a d i, M., S. P a r s a. Natural Language Requirements Testability Measurement Based on Requirement Smells. – Neural Computing and Applications, 2024, pp. 1-35.
50. D o c h e v, D., I. H r i s t o v. On-the-Job e-Training – from Requirements to Design 1. – Cybernetics and Information Technologies, Vol. **3**, 2003, No 2, pp. 45-54.
51. M a y v a n, B. B., A. R a s o o l z a d e g a n, A. J. J a f a r i. Bad Smell Detection Using Quality Metrics and Refactoring Opportunities. – Software: Evolution and Process, 2020, No December 2019, pp. 1-33. DOI: 10.1002/smr.2255.
52. O o, K. H., A. N o r d i n, A. R. I s m a i l, S. S u l a i m a n. An Analysis of Ambiguity Detection Techniques for Software Requirements Specification (SRS). – International Journal of Engineering & Technology, Vol. **7**, 2018, pp. 501-505. DOI: 10.14419/ijet.v7i2.29.13808.
53. F e r r a r i, A., G. G o r i, B. R o s a d i n i, I. T r o t t a, S. B a c h e r i n i, A. F a n t e c h i, S. G n e s i. Detecting Requirements Defects with NLP Patterns: An Industrial Experience in the Railway Domain. – Empirical Software Engineering, Vol. **23**, 2018, No 6, pp. 3684-3733. DOI: 10.1007/s10664-018-9596-7.

54. E z z i n i, S., S. A b u a l h a i j a, C. A r o r a, M. S a b e t z a d e h, L. C. B r i a n d. Using Domain-Specific Corpora for Improved Handling of Ambiguity in Requirements. – In: Proc. of International Conference on Software Engineering, 2021, pp. 1485-1497. DOI: 10.1109/ICSE43902.2021.00133.
55. O s a m a, M., A. Z a k i-I s m a i l, M. A b d e l r a z e k, J. G r u n d y, A. I b r a h i m. Score-Based Automatic Detection and Resolution of Syntactic Ambiguity in Natural Language Requirements. – In: Proc. of IEEE International Conference on Software Maintenance and Evolution (ICSME'20), 2020, pp. 651-661. DOI: 10.1109/ICSME46990.2020.00067.
56. T h i t i s a t h i e n k u l, P., N. P r o m p o o n. Quality Assessment Method for Software Requirements Specifications Based on Document Characteristics and Its Structure. – In: Proc. of 2nd International Conference on Trustworthy Systems and Their Applications (TSA'15), 2015, pp. 51-60. DOI: 10.1109/TSA.2015.19.
57. G l e i c h, B., O. C r e i g h t o n, L. K o f. Ambiguity Detection: Towards a Tool Explaining Ambiguity Sources. – Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol. **6182 LNCS**, 2010, No May, pp. 218-232. DOI: 10.1007/978-3-642-14192-8_20.
58. W i l m i n k, M., C. B o c k i s c h. On the Ability of Lightweight Checks to Detect Ambiguity in Requirements Documentation. – Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol. **10153 LNCS**, 2017, pp. 327-343. DOI: 10.1007/978-3-319-54045-0_23.
59. T j o n g, S. F., D. M. B e r r y. The Design of SREE – A Prototype Potential Ambiguity Finder for Requirements Specifications and Lessons Learned. – Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol. **7830 LNCS**, 2013, pp. 80-95. DOI: 10.1007/978-3-642-37422-7_6.
60. K a t o, T., K. T s u d a. A Method of Ambiguity Detection in Requirement Specifications by Using a Knowledge Dictionary. – Procedia Computer Science, Vol. **207**, 2022, pp. 1482-1489. DOI: 10.1016/j.procs.2022.09.205.
61. M o r e n o, V., G. G. O r c i d, E. P a r r a, A. F r a g a. Application of Machine Learning Techniques to the Flexible Assessment and Improvement of Requirements Quality. 2020. DOI: 10.1007/s11219-020-09511-4.
62. H a n i s c h, L. Detecting Vague Requirements with Machine Learning Detecting Vague Requirements with Machine Learning Detektion von Vagen Anforderungen mit Maschinellem Lernen. – Department of Informatics Technical University of Munich, 2020.
63. O s m a n, M. H., M. F. Z a h a r i n. Ambiguous Software Requirement Specification Detection: An Automated Approach. – In: Proc. of International Conference on Software Engineering, 2018, pp. 33-40. DOI: 10.1145/3195538.3195545.
64. Y a n g, H., A. de R o e c k, V. G e r v a s i, A. W i l l i s, B. N u s e i b e h. Analysing Anaphoric Ambiguity in Natural Language Requirements. – Requirements Engineering, Vol. **16**, 2011, No 3, pp. 163-169. DOI: 10.1007/s00766-011-0119-y.
65. E z z i n i, S., S. A b u a l h a i j a, C. A r o r a, M. S a b e t z a d e h. Automated Handling of Anaphoric Ambiguity in Requirements: A Multi-Solution Study. – Proceedings International Conference on Software Engineering, Vol. **2022-May**, 2022, pp. 187-199. DOI: 10.1145/3510003.3510157.
66. Z h a n g, H., M. A l i, P. T e l l. Identifying Relevant Studies in Software Engineering. – Information and Software Technology, Vol. **53**, 2011, No 6, pp. 625-637. DOI: 10.1016/j.infsof.2010.12.010.
67. W o h l i n, C. Guidelines for Snowballing in Systematic Literature. – Studies and a Replication in Software Engineering, 2014. DOI: 10.1145/2601248.2601268.
68. C r u z e s, D. S., T. D y b å. Recommended Steps for Thematic Synthesis in Software Engineering. – In: Proc. of International Symposium on Empirical Software Engineering and Measurement, 2011, No 7491, pp. 275-284. DOI: 10.1109/esem.2011.36.
69. Y a n g, B., Z. X i n g, X. X i a, C. C h e n, D. Y e, S. L i. UIS-Hunter: Detecting UI Design Smells in Android Apps. – Proceedings International Conference on Software Engineering, Vol. **1**, 2021, No c, pp. 89-92. DOI: 10.1109/ICSE-Companion52605.2021.00043.
70. J u n k e r, M., S. E d e r, L. H e i n e m a n n, C. G m b h, R. V a a s, P. B r a u n, V. A g. Hunting for Smells in Natural Language. – Tests No 1, pp. 4-7. DOI: 10.1109/ICSE.2013.6606682.

71. Z h a o, L., W. A l h o s h a n, A. F e r r a r i, K. J. L e t s h o l o, M. A. A j a g b e, E. V. C h i o a s c a, R. T. B a t i s t a-N a v a r r o. Natural Language Processing (NLP) for Requirements Engineering : A Systematic Mapping Study. – ACM Computing Surveys (CSUR), Vol. **54**, 2021, No 3, pp. 1-41. DOI: 10.1145/3444689.
72. K u m a w a t, D., V. J a i n. POS Tagging Approaches : A Comparison. – International Journal of Computer Applications, Vol. **118**, 2015, No 6, pp. 32-38. DOI: 10.5120/20752-3148.
73. A l t i n o k, D. Mastering SpaCy – An End-to-End Practical Guide to Implementing NLP Applications Using the Python Ecosystem. Packt Publishing, Ltd., 2021.
74. T o u t a n o v a, K., D. K l e i n, C. D. M a n n i n g. Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. – In: Proc. of HLT-NAACL 2003, 2003, No June, pp. 173-180. DOI: 10.3115/1073445.1073478.
75. G a r d n e r, M., J. G r u s, M. N e u m a n n, O. T a f j o r d, P. D a s i g i, N. F. L i u, M. P e t e r s, M. S c h m i t z, L. Z e t t l e m o y e r. AllenNLP: A Deep Semantic Natural Language Processing Platform. – In: Proc. of Workshop for NLP Open Source Software (NLP-OSS'17), 2017, pp. 3-8. DOI: 10.18653/v1/W18-2501.
76. E c k a r t, R., D. C. I r y n a. A Broad-Coverage Collection of Portable NLP Components for Building Shareable Analysis Pipelines. – Proceedings of Workshop on Open Infrastructures and Analysis Frameworks for HLT, Vol. **2**, 2014, No 1, pp. 1-11.
77. C u n n i n g h a m, H., S. C o m p u t e r s, N. M a y, H. C u n n i n g h a m. GATE, a General Architecture for Text Engineering GATE, a General Architecture for Text Engineering. – Computers and the Humanities, Vol. **36**, 2002, No 2, pp. 223-254. DOI: 10.1023/A:1014348124664.
78. T s u r u o k a, Y., Y. T a t e i s h i, J. D. K i m, T. O h t a, J. M c N a u g h t, S. A n a n i a d o u, J. T s u j i i. Developing a Robust Part-of-Speech Tagger for Biomedical Text. – Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol. **3746 LNCS**, 2005, pp. 382-392. DOI: 10.1007/11573036_36.
79. G i u l i a n o, C. jWeb1T : A Library for Searching the Web 1T 5. – Gram Corpus, 2011, pp. 2011.
80. Z e s c h, T., C. M ü l l e r, I. G u r e v y c h. Extracting Lexical Semantic Knowledge from Wikipedia and Wiktionary. – LREC, Vol. **8**, 2007, No 2008, pp. 1646-1652.
81. Y a o, J. Automated Sentiment Analysis of Text Data with Automated Sentiment Analysis of Text Data with NLTK. 2019, pp. 0-8. DOI: 10.1088/1742-6596/1187/5/052020.
82. L o p e r, E., S. B i r d. NLTK: The Natural Language Toolkit. ArXiv Preprint Cs/0205028, 2002.
83. H o r n b y, A. S. Oxford Advanced Learner's Dictionary of Current English. – Oxford Univer Press.
84. F a b b r i n i, F., M. F u s a n i, S. G n e s i, G. L a m i. An Automatic Quality Evaluation for Natural Language Requirements. – In: Proc. of 7th International Workshop on Requirements Engineering: Foundation for Software Quality REFSQ, 2001, No March 2014.
85. L a m i, G., S. G n e s i, F. F a b b r i n i, M. F u s a n i, G. T r e n t a n n i. An Automatic Tool for the Analysis of Natural Language Requirements. – Informe Técnico, CNR Information Science and Technology Institute, 2004.
86. V e r m a, T a n u G a u r D. Tokenization and Filtering Process in RapidMiner. – International Journal of Applied Information Systems, Vol. **7**, 2014, No 2, pp.16-18.
87. R e a d, J., R. D r i d a n, S. O e p e n, L. J. S o l b e r g. Sentence Boundary Detection: A Long Solved Problem ? – Proceedings of COLING 2012: Posters. No December 2012, pp. 985-994.
88. Q i, P., Y. Z h a n g, Y. Z h a n g, J. B o l t o n, C. D. M a n n i n g. Stanza: A Python Natural Language Processing Toolkit for Many Human Languages. – ArXiv Preprint ArXiv:200307082, 2020.
89. A s a n o, K., S. H a y a c h, M. S a e k i. Detecting Bad Smells of Refinement in Goal-Oriented Requirements Analysis. – Advances in Conceptual Modeling. – In: Proc. of ER 2017 Workshops AHA, MoBiD, MREBA, OntoCom, and QMMQ, 2017, pp. 122-132. DOI: 10.1007/978-3-319-70625-2.
90. M a n n i n g, C. D., M. S u r d e a n u, J. B a u e r, J. F i n k e l, S. J. B e t h a r d, D. M c C l o s k y. The Stanford CoreNLP Natural Language Processing Toolkit. – In: Proc. of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, 2014, pp. 55-60.
91. M i l l e r, G. A. WordNet : A Lexical Database for English. – Communications of the ACM, Vol. **38**, 1995, No 11, pp. 39-41. DOI: 10.1145/219717.219748.

92. H i d e k i, S h i m a. WordNet Similarity for Java Relatedness – Similarity Algorithms, pp. 2-7.
93. B r i a n d, L., M. A l f e r e z, D. T o r r e, M. S a b e t z a d e h. On Systematically Building a Controlled Natural. – Empirical Software Engineering, 2021. DOI: 10.1007/s10664-021-09956-6.
94. D e i s s e n b o e c k, F., L. H e i n e m a n n, B. H u m m e l, E. J u e r g e n s. Flexible Architecture Conformance Assessment with ConQAT. – ICSE'10, 2010, No 1.
95. W i l s o n, W. M., W. M. W i l s o n, L. H. R o s e n b e r g, L. R o s e n b e r g, L. E. H y a t t. Automated Analysis of Requirement Specifications. – In: Proc. of 19th International Conference on Software Engineering, 1997, pp. 161-171. DOI: 10.1145/253228.253258.
96. P e d r e g o s a, F., G. V a r o q u a u x, A. G r a m f o r t, et al. Scikit-Learn: Machine Learning in Python. – Journal of Machine Learning Research, Vol. **12**, 2011, pp. 2825-2830.
97. H a l l, M., H. N a t i o n a l, E. F r a n k, G. H o l m e s, B. P f a h r i n g e r, P. R e u t e m a n n, I. H. W i t t e n. The WEKA Data Mining Software. – An Update, Vol. **11**, No 1, pp. 10-18. DOI: 10.1145/1656274.1656278.
98. L e v y, R., G. A n d r e w. Tregex and Tsurgeon: Tools for Querying and Manipulating Tree Data Structures. – LREC, 2005.
99. Z h o u, X., S. L i. A Map of Threats to Validity of Systematic. – Literature Reviews in Software Engineering, 2016, pp. 153-160. DOI: 10.1109/APSEC.2016.62.
100. A m p a t z o g l o u, A., S. B i b i, P. A v g e r i o u, M. V e r b e e k, A. C h a t z i g e o r g i o u. Identifying, Categorizing and Mitigating Threats to Validity in Software Engineering Secondary Studies. 2016. DOI: 10.1016/j.infsof.2018.10.006.
101. S t a p i c, Z., E. G. L ó p e z, A. G. C a b o t, L. O r t e g a de M., V. S t r a h o n j a. Performing Systematic literature Review in Software Engineering. – In: Proc. of Central European Conference on Information and Intelligent Systems, 2012, No 2012, pp. 442-493. DOI: 10.1145/1134285.1134500.
102. W o h l i n, C., P. R u n e s o n, M a r t i n Ḧ o s t, M. C. O h l s s o n, B. R e g n e l l, A. W e s s l´e n. Experimentation in Software Engineering. Springer Science & Business Media, 2012.
103. F a r r u g i a, P., B. A. P e t r i s o r, F. F a r r o k h y a r, M. B h a n d a r i. Practical Tips for Surgical Research: Research Questions, Hypotheses and Objectives. – Canadian Journal of Surgery Journal Canadien de Chirurgie, Vol. **53**, 2010, No 4, pp. 278-281.
104. B e e r, A., M. F e l d e r e r. Initial Investigations on the Influence of Requirement Smells on Test-Case Design. – In: Proc. of 25th IEEE International Requirements Engineering Conference Workshops (REW'17), 2017, pp. 323-326. DOI: 10.1109/REW.2017.43.
105. P e i, Z., L. L i u, C. W a n g, J. W a n g. Requirements Engineering for Machine Learning : A Review and Reflection. – In: Proc. of 30th IEEE International Requirements Engineering Conference Workshops (REW'22), 2022, pp. 166-175. DOI: 10.1109/REW56159.2022.00039.
106. R a t h, M., P. R e m p e l, M. P a t r i c k. The IlmSeven Dataset. – In: Proc. of 25th IEEE International Requirements Engineering Conference, 2017, pp. 516-519.
107. S o m m e r v i l l e, I. Software Engineering. Ninth Edit. Addison-Wesley, 2011.
108. A v e r s a n o, L., U. C a r p e n i t o, M. I a m m a r i n o. An Empirical Study on the Evolution of Test Smell. – In: Proc. of 42nd ACM/IEEE International Conference on Software Engineering: Companion, ICSE-Companion, 2020, pp. 149-151. DOI: 10.1145/3377812.3382176.

Received: 03.10.2024; Accepted: 23.10.2024